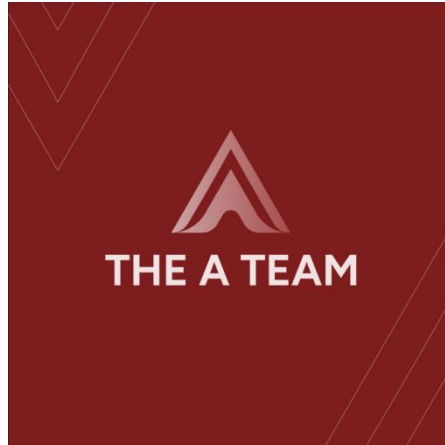


University of Oklahoma

Sonner Tire Company



The A Team

[Ben Finley, Brandon Niver, Meg Marrs, Presley Lindsay, Roshan Rai]

MIS 3353 – Database Management

Prof. Swetha Siripurapu

Executive Summary

The first section of this project will outline the members of the A Team as well as personal information about each one of them. Following that is the first major section of the project, which consists of our conceptual design. This section includes the information gathered during the client meeting as well as what the A Team learned following the Q&A portion. Following the conceptual design section, we state what an ERD is and what we stand from using them as well as assumptions our team made while constructing our ERD. All of this information is found in the significant assumptions section. In addition, we also list the business cycles used and why we used them. Following that is where our team displays our personalized ERD for Sonner Tires, as well as the changes made to its original design.

After the ERD creation section, the logical design section displays the logical design created from the ERD as well as the normalization of the relations. The next section displays the differences between an ERD and the created normalized relations as well as what referential integrity is. Following this section, the A Team created the physical design section, which consists of the data dictionary used, denormalization, the actual implementation of the physical design, statements on what challenged the team and how we addressed them, and lastly, what strength and weaknesses our team encountered during the implementation phase. The next section is where our team lists the SQL statements requested as well as additional queries created by us.

The second to last section consists of the user documentation, which explains instructions on how to use the database. Following the instructions, there are statements from the team about what we all learned throughout this project. The last section, the appendix, consists of an image of the A Team's contract, our data dictionary model, and the project management information. The project management information displays the hours of work accumulated from each team member for each section as well as the final cost of the project, which totaled up to \$1,217.

Table of Contents

Executive Summary	2
Table of Contents	3
Get to Know the Team: The A Team	4
Conceptual Design	5
The Client Meeting.....	5
Q&A During the Meeting & Information We Learned	5
Significant Assumptions	7
What is an ERD? Why is it necessary?	7
Business Cycles Used.....	8
ERD Created	9
Breaking Down the Data Provided by Client	9
Changed made to generic ERDs	11
Logical Design	12
Normalization	12
Normalization of Data Provided by the Client	12
Normalized Relations	14
Differences between ERD and Normalized Relations	17
Referential Integrity.....	17
Physical Design and Implementation	19
Data Dictionary	19
Denormalization	19
Implemented Physical Design	19
Challenges Faced/Addressed During Implementation	20
Strengths and Weaknesses Encountered During Implementation	21
Specific SQL Statements Requested	22
Three Additional Queries	27
User Documentation	30
Lessons Learned	34
Team Function/Insights.....	34
Appendix.....	35
Team Contract	35
Data Dictionary Model	36
Project Management	40

Get to Know the Team: The A Team

Benjamin Finley



Major: MIS

Year: Sophomore

Internship Experience: None

Background: I am originally from Frisco, Texas and wasn't an MIS major when coming to OU initially but switched in Fall 2022. I also want to pursue the MS degree in Management Information Technology offered at OU.

Brandon Niver



Major: MIS

Year: Junior

Internship Experience: Internship with OneGas' IT department this summer.

Background: A lifelong Oklahoman that always dreamed of coming to OU. I wasn't a business major coming into college but have no regrets since switching.

Roshan Rai



Major: MIS

Year: Junior

Internship Experience: None

Background: I am an international student. I am from Nepal. I am majoring in MIS and minoring in Computer Science.

Margaret (Meg) Marrs



Major: MIS, Accounting

Year: Sophomore

Internship Experience: None

Background: Born and raised in Denver, Colorado, and came to OU to experience college an environment that cares so deeply about the community.

Presley Lindsay



Major: Marketing

Year: Junior

Internship Experience: I had an internship with Movement Mortgage in Social Media Marketing last summer, and I currently have an internship in the same division with Fairway Independent Mortgage Company.

Background: I am from The Woodlands, Texas, and I came to OU to find the happy medium between close enough to home to drive while still feeling far enough away. I recently dropped my MIS major, but I am still enjoying this class!

Conceptual Design

Conceptual Design is an important step in the creation of a database where entities and relationships, and attributes are identified. Then, they are put into a visual representation in an ERD. It's like looking at the simple plan of a house. You get a general idea of what it will look like, but not the specific details.

The Client Meeting

For this project, our team was required to join a group conference call to talk about questions we had concerning specific elements needed to form a correct ERD model for Sonner Tire. This meeting took place on Zoom on March 6, 2023, at 11:00 A.M. The interviewers consisted of all the members of *The A Team*: Benjamin Finley, Brandon Niver, Roshan Rai, Meg Marrs, and Presley Lindsay. The interviewee for this call was Swetha Siripurapu. A bulleted list is listed below for a more condensed version of this information.

- Meeting Time: 11:00 A.M. on 3/6/2023
- Length: ~ 10 minutes
- Location: Zoom
- Interviewers: Benjamin Finley, Brandon Niver, Roshan Rai, Meg Marrs, Presley Lindsay
- Interviewee: Swetha Siripurapu

Q&A During the Meeting & Information We Learned

Name	Question(s)	Answer(s)
Benjamin Finley	1. Is the "number" in "The number and type of job performed by each of our employees" meant to be a JobID? Do you want to see the connection between which employees completed which job and the type of job?	1. Yes, JobID is to track each separate job completed by the firm and should trace to each employee that participated
Brandon Niver	1. When stating if customers are best/problematic, does that come down to solely not paying an amount on time or paying the whole sum later in the payment process? 2. Are we looking to track all employees or more specific ones? 3. Are the highest selling categories split up into two categories (bulk/single order)?	1. Mainly not paying on time, so we need to make an assumption 2. Any employee 3. Just in general
Roshan Rai	1. Is the insurance offered by Sonner tire or the tire manufacturer? 2. Can the same Tire be sold by multiple manufacturers?	1. Sonner tire covers the insurance 2. Yes

Meg Marrs	<ol style="list-style-type: none"> 3. Does a customer who drops is responsible for pick up? 1. How is marketing material sent? Do you have access to the customer's home address, email address, phone number? 2. employees (paid hourly or salary) 3. do they have managers, if so, how many for the location 	<ol style="list-style-type: none"> 3. We can set it up anyway 1. Tracking some type of address 2. Up to us (does not need to be tracked) 3. No managers
Presley Lindsay	<ol style="list-style-type: none"> 1. How many suppliers do Sooner Tires get stock from? 2. About how many employees do each job? 	<ol style="list-style-type: none"> 1. Varies between five and ten 2. One

Significant Assumptions

This section's purpose is to describe the assumptions we made about the company operations to make the ERD. An assumption is a statement about the ERD is not based on certain information. These are used to make a more complete model for Sonner Tires and fill in any gaps of missing or unconfirmed information. Some of our assumptions are listed below:

1. The customer who dropped the car may or may not pick up the car, meaning, anyone can pick up the cars that are not owned by them. However, in the event, the owner or the person who dropped the car does not come to pick it up, we verify the information of the pickup customer by calling the owner and emailing them while also storing the information of the person who picked the car
2. The business can take partial shipments of the tires that purchase in bulk. This is portrayed in the relationship between the TireVendor and the <PurchaseOrderLine> tables. If it was connected to the PurchaseOrder table, then the company would only take complete shipments.
3. Employees are paid by the hour instead of by salary. This is shown in the Employee table. This is a simple way to include more precise information about the employee that would be useful to the business.
4. For the Employee table, there are no managers. This is shown by a lack of a ManagerID. This makes it easier to track employees and makes sense because it is a small business. Also, given we don't have any positional distinction, we pay our employees the same salary hourly.
5. When adding a Discount to the Database, we must put the Active and InActive Date for that Discount so that it automatically expires when the date is past. This is done to prevent people using the same discount again and again in the event we forget to put the InActiveDate for the Discount.
6. The Employee who receives the Tire from the vendor must also verify and approve the Tires based on the Purchase order and Invoice.
7. We store the Tracking Number and the carrier once issued by our vendors for specific Purchase order to check the status of the order. However, we do not deliver tires to the Customer (No relationship between Delivery and SalesOrder or SalesOrderLine).
8. We only sell either Tire or Service to those Tires (Total Partialization). Also, we sell Tire and Service separately to customers (Disjoint). However, when Customer buys tire from us, we provide them free servicing (especially for certain period of time). In this event, we charge them for Tire and Service separately but apply services with no charge. For example, PROD037 is one of the services in product table that we apply for free if a customer buys tires from Sonner Tire. (This is to prevent complications with Overlap of Tire and Service).

What is an ERD? Why is it necessary?

An ERD, or an Entity Relationship Diagram, is a graphical model showing the connections between various entities and how they relate to one another. They are mostly used in databases to easily portray the structure of a database. This visualization of the database makes it so that all relationships between entities can be easily seen and understood.

The easiest way to think about an ERD is to think about it as a blueprint of a house. Without the blueprint for a house, it is nearly impossible to create everything needed or wanted. This is why

it is important that we use an ERD for this specific project. To make sure that we accomplish, to the fullest extent, everything that Sonner Tire wants us to do it is essential to have a fully developed and functional ERD.

Business Cycles Used

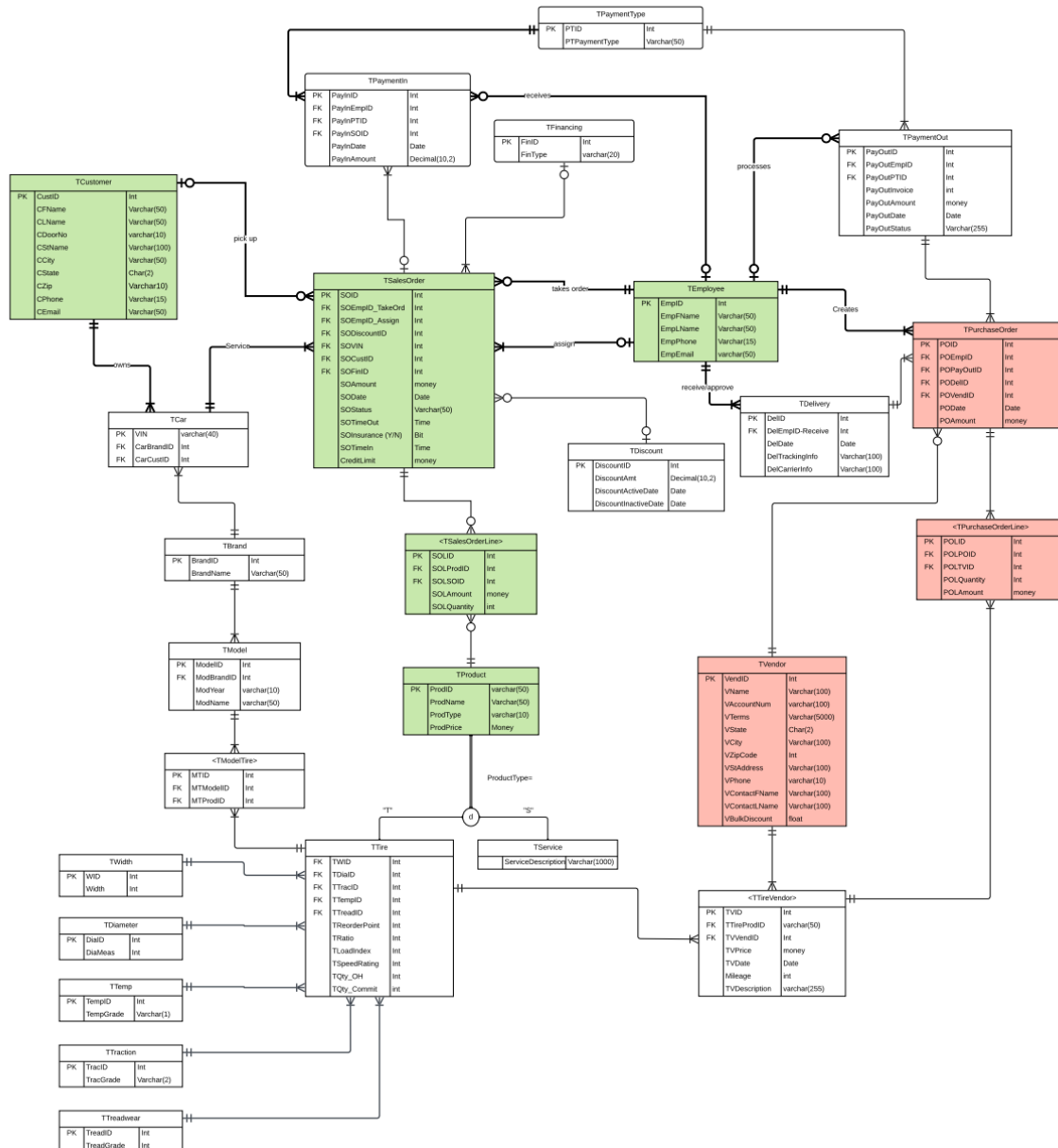
After meeting with the client, going through the company information and reviewing the sample data collected by Sonner Tire, we identified that the client's business involves purchasing tires from vendors and selling them to their customers. As the company is not producing anything. The Production Cycle is not relevant to their operations, and only the Expenditure and Revenue Cycles are necessary. We incorporated the Revenue Cycle into the Entity Relationship Diagram (ERD) because the client's primary objective is to generate profit by selling products.

Additionally, the Expenditure Cycle is essential to the ERD because Sonner Tire needs to purchase tires from vendors to sustain its business activities. The Expenditure Cycle encompasses all activities related to the acquisition of resources necessary for the operation of a business. This cycle includes the processes of requesting, receiving, and paying for goods and services (Tire). On the other hand, the Revenue Cycle focuses on the processes involved in selling goods or services to customers, such as order processing, delivery, and invoicing. In the case of Sonner Tire, the business revolves around purchasing tires from vendors and selling them to customers. Therefore, the Revenue Cycle is crucial in facilitating sales transactions and generating revenue. The Expenditure Cycle is also critical in managing the procurement of goods from vendors to ensure that the business has the necessary inventory to meet customer demand.

In summary, the ERD for Sonner Tire does not include the Production Cycle because the client does not produce anything. Instead, we have incorporated the Expenditure and Revenue Cycles into the ERD to capture the essential business processes involved in purchasing and selling tires. This approach allows us to develop an accurate representation of the client's business operations and facilitate effective management of their resources.

ERD Created

This section includes an image of the ERD we created to complete the tasks requested by Sonner Tire. Below the image is a link that takes you to our LucidChart document, that way it is easily accessible if there is a problem with the image or if you cannot see it clearly enough.



Link to ERD: https://lucid.app/lucidchart/eac3519a-b74c-49b1-bdba-14dc6c44f6e7/edit?page=0_0&invitationId=inv_feafca82-c076-491e-a43a-e965dc5ee5eb#

Breaking Down the Data Provided by Client

We received sample data of the company from our client, which consisted of two tables. By analyzing this data, we were able to determine the necessary entities for our Entity Relationship Diagram (ERD). The figure below displays the couple of records of the tables provided by the

Tire Name	Manufacturer	Good for	Mileage	Description	Cost	Sale Price (without taxes)
235/55R17	BFGoodrich	Ford Escape 2014	50000	Defy the weather, control the curves, and enjoy the drive for miles and miles	\$ 88.50	\$ 146.99
235/55R17	BFGoodrich	Ford Escape 2014	45000	BFGoodrich's Best Ultra-High Performance All-Season Tire Ever.	\$ 80.25	\$ 131.99
235/55R17	G-Force R1	Ford Escape 2015	60000	Defy the weather, control the curves, and enjoy the drive for miles and miles	\$88.50	\$146.99
235/55R17	G-Force Rival	Ford Escape 2015	45000	BFGoodrich's Best Ultra-High Performance All-Season Tire Ever.	\$80.25	\$131.99
235/55R17	SafeContact LX	Ford F-150 2017	60000	Continental's tire built for greater good	\$130.50	\$218.00
235/55R17	Pilot Sport 3	Ford F-150 2017	45000	Quiet, comfortable ride	\$87.25	\$148.99
235/55R17	Ecopia EP20	Ford F-150 2018	45000	Armed with technologies that help deliver durability with a good tread life	\$93.25	\$158.99



Changed made to generic ERDs

This section includes an image of the original ERD used, an updated ERD, and the specific changes that we made to the original ERD. We clearly state explanations as to why we made the changes that we have made to the original ERD in that specific column of this section.

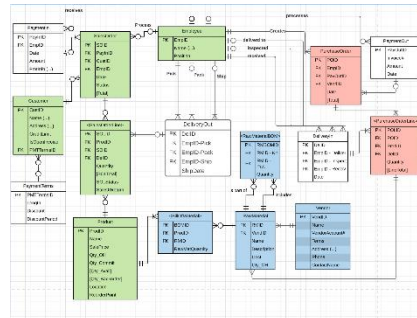
Change

Explain the change in 3-5 sentences

We deleted the production cycle entirely because it's not applicable to the company. We created a purchase history for customers and changed the DeliveryIn table to the Job table to track employees to jobs they worked on. We also moved the Vendor table to the expenditure cycle because they are spending money on buying tires for their business. Lastly, we added, subtracted, and changed many different attributes to better fit the company.

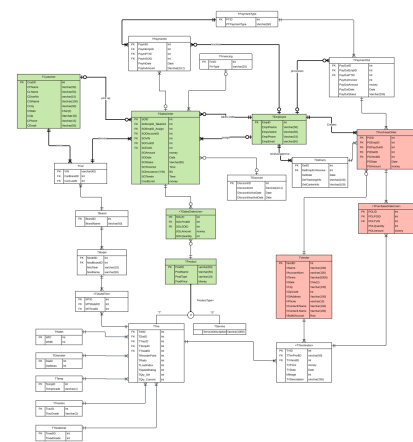
Original ERD

Show picture



Updated ERD

Show picture



Logical Design

Logical Design is the process of refining the conceptual design of an ERD. Logical design involves normalizing the diagram and making sure it is atomic. This increases the integrity of the data and creates a more precise model than the conceptual design. This step is like looking at a more detailed blueprint than the simple blueprint of the conceptual design step.

Normalization

Normalization is a technique used to organize data in a structured way to reduce data redundancy and improve data integrity. It involves breaking down a database table into smaller, more specialized tables and attributes, referred to as atomicity. The primary goal of normalization is to minimize data duplication, which can lead to inconsistencies and errors in the database. By breaking a table into smaller tables, the data can be organized in a more efficient manner, making it easier to search and manipulate. By normalizing, the data is organized in a way that makes it easier to manage, maintain, and query. There are multiple normal forms (NF) that a normalized ERD can be in 0NF, 1NF, 2NF, and 3NF. Going in order, 3NF is the most normalized level.

- 0NF is where the data is not atomic
- 1NF is where the data has Partial Functional Dependencies (PFDs) and the data is atomic
- 2NF is atomic, does not have PFDs, has Functional Dependencies (FDs), and Transitive Dependencies (TDs)
- 3NF is atomic, has FDs, and no TDs

Pertaining to assisting Sooner Tire Company, normalization ensures that the database is well-structured while preventing data inconsistencies that could appear if the database was not normalized properly. To avoid these potential problems, the data would be stored in one place allowing it to be easily maintained. Normalization also allows the relationship between multiple tables such as customers, orders, and inventory to be well-defined.

Normalization of Data Provided by the Client

We were given two tables of sample data by our client. These are the first 5 records of sample data provided by our client. We realized that this data is not in the highest Normal Form (3NF) that we would like it to be to be helpful for our database. Therefore, in this section, we will be normalizing these tables into 3NF as we learned in class. To see the purpose and Normalization, please refer to the previous section. Note that Single Underline means the attribute is Primary Key and Double underline means the attribute is Foreign Key.

Car Manufacture	Model	Tire
Ford	Escape 2019, 2018, 2017, 2016, 2015, 2014	235/55R17, 235/50R18, 235/50R19, 235/45R19
Ford	Mustang 2019, 2018, 2017, 2016, 2015, 2014	235/50R18, 255/40R19
Ford	Edge 2019, 2018, 2017, 2016, 2015, 2014	245/50R20, 265/40R21
Ford	Explorer 2019, 2018, 2017, 2016, 2015, 2014	255/50R20, 245/60R18, 235/65R18
Honda	Accord 2018, 2017, 2016, 2015, 2014	235/40R19, 225/50R17, 215/55R17

Normal Form: 0NF

Explanation: It is non-atomic because there are attributes with multiple records of data in them. If the data is not atomic, then it cannot be anything greater than 0NF.

Normalizing it to 3NFTBrand(BrandID, BrandName)TModel(ModelID, ModelName, Year, MBrandID)

Foreign key MBrandID references TBrand

Not Null

On Delete Restrict

TTire(TireID, TireName)

TModelTire(TMID, TMTireID, TMMModelID)

Foreign key TMTireID references TTire

Not Null

On Delete Restrict

Foreign key TMMModelID references TModel

Not Null

On Delete Restrict

Tire Name	Manufacturer	Good for	Milage	Description	Cost	Sale Price (without taxes)
235/55R17	BFGoodrich	Ford Escape 2014	60000	Defy the weather, control the curves, and enjoy the drive for miles and miles	\$ 88.50	\$ 146.99
235/55ZR17	BFGoodrich	Ford Escape 2014	45000	BFGoodrich's Best Ultra-High Performance All-Season Tire Ever.	\$ 80.25	\$ 131.99
235/55R17	G-Force R1	Ford Escape 2015	60000	Defy the weather, control the curves, and enjoy the drive for miles and miles	\$88.50	\$146.99
235/55R17	G-Force Rival	Ford Escape 2016	45000	BFGoodrich's Best Ultra-High Performance All-Season Tire Ever.	\$80.25	\$131.99
235/55R17	SureContact LX	Ford F-150 2017	60000	Continental's tire built for greater good	\$130.50	\$218.00

Normal Form: ONF

Explanation: The good for attribute is not atomic.

Normalizing it to 3NF

(Assumption: Sales Price is a Derived attribute based on the given data as Sales Price seems to be calculated based on the Cost. Therefore, we have not mentioned in in the following normalized relations)

TManufacturer(ManufacturerID, ManufacturerName)TBrand(BrandID, BrandName)TModel(ModelID, ModelName, Year, MBrandID)

Foreign key MBrandID references TBrand

Not Null

On Delete Restrict

TTire(TireID, TireName)

TModelTire(TMID, TMTireID, TMMModelID)

Foreign key TMTireID references TTire

Not Null

On Delete Restrict

Foreign key TMMModelID references TModel

Not Null

On Delete Restrict

TTireManufacturer(TireManuID, TireManuTireID, TireManuManucaturerID, Description, Mileage, Cost)

Foreign key TireManuTireID references TTire

Not Null

On Delete Restrict

Foreign key TireManuManucaturerID references TManufacturer

Not Null

On Delete Restrict

Normalized Relations

PK = underline

FK = *

TCustomer(CustID, CFName, CLname, CDoorNo, CStName, CCity, CState, CZip, CPhone, CEmail)

TVendor(VendID, VName, VAccountNum, VTerms, VState, VCity, VZipCode, VStAddress, VPhone, VContactFName, VContactLName, VBulkDiscount)

TEmployee(EmpID, EmpFName, EmpLName, EmpPhone, EmpEmail)

TDiscount(DiscountID, DiscountAmt, DiscountActiveDate, DiscountInactiveDate)

TBrand(BrandID, BrandName)

TCar(VIN, CarBrandID*, CarCustID*)

Foreign key CarBrandID references TBrand

Not Null

On Delete Restrict

Foreign key CarCustID references TCustomer

Not Null

On Delete Restrict

TFinancing(FinID, FinType)

TSalesOrder(SOID, SOEmpID-TakeOrd*, SOEmpID-Assign*, SODiscountID*, SOVIN*, SOCustID*, SOFinID*, SOAmount, SODate, SOSStatus, SOTimeOut, SOInsurance(Y/N), SOTimeIn)

Foreign key SOEmpID-TakeOrd references TEmployee

Not Null

On Delete Restrict

Foreign key SOEmpID-Assign references TEmployee

Not Null

On Delete Restrict

Foreign key SODiscountID references TDiscount

Null Allowed

On Delete Set Null
Foreign key SOVIN references TCar
Not Null
On Delete Restrict
Foreign key SOCustID references TCustomer
Null Allowed
On Delete Set Null
Foreign key SOFinID references TFinancing
Null Allowed
On Delete Set Null

TPaymentType(PTID, PITPaymentType)

TPaymentIn(PayInID, PayInEmpID*, PayInPTID*, PayInSOID*, PayInDate, PayInAmount)
Foreign key PayInEmpID references TEmployee
Null Allowed
On Delete Set Null
Foreign key PayInPTID references TPaymentInType
Not Null
On Delete Restrict
Foreign key PayInSOID references TSalesOrder
Null Allowed
On Delete Set Null

TPaymentOut(PayOutID, PayOutEmpID*, PayOutPTID*, PayOutInvoice, PayOutAmount, PayOutDate, PayOutStatus)
Foreign key PayOutEmpID references TEmployee
Null Allowed
On Delete Set Null
Foreign key PayOutPOTID references TPaymentOutType
Not Null
On Delete Restrict

TProduct(ProdID, ProdName, ProdType, ProdPrice

TWidth(WID, WMeas)

TDiameter(DialID, DiaMeas)

TTemp(TempID, TempGrade)

TTraction(TracID, TracGrade)

TTreadwear(TreadID, TreadGrade)

TTire(TireProdID*, TWID*, TDialID*, TTracID*, TTempID*, TTreadID*, TReorderPoint, TTRatio, TLoadIndex, TSpeedRating, TQty OH, TQty_Commit)
Foreign key TireProdID references TProduct

Not Null
On Delete Restrict
Foreign key TWID references TWidth
Not Null
On Delete Restrict
Foreign key TDia references TDiameter
Not Null
On Delete Restrict
Foreign key TTracID references TTraction
Not Null
On Delete Restrict
Foreign key TTempID references TTemp
Not Null
On Delete Restrict
Foreign key TTreadID references TTreadwear
Not Null
On Delete Restrict

TService(ServiceProdID*, ServiceDescription)
Foreign key ServiceProdID references TProduct
Not Null
On Delete Set Null

TSalesOrderLine(SOLID, SOLProdID*, SOLSOID*, SOLQuantity, SOLAmount)
Foreign key SOLProdID references TProduct
Not Null
On Delete Restrict
Foreign key SOLSOID references TSalesOrder
Not Null
On Delete Restrict

TModel(ModelID, ModBrandID*, ModName, ModYear)
Foreign key ModBrandID reference TBrand
Not Null
On Delete Restrict

TModelTire(MTID, MTModelID*, MTProdID*)
Foreign key MTModelID reference TModel
Not Null
On Delete Restrict
Foreign key MTTireProdID references TTire
Not Null
On Delete Restrict

TTireVendor(TVID, TVTireProdID*, TVVendID*, TVPrice, TVDate)
Foreign key TVTireProdID references TProduct
Not Null
On Delete Restrict

Foreign key TVVendID references TVendor
Not Null
On Delete Restrict

TDelivery(DelID, DelEmpID-Receive*, DelDate, DelTrackingInfo, DelCarrierInfo)
Foreign key DelEmpID-Receive references TEmployee
Not Null
On Delete Restrict

TPurchaseOrder(POID, POEmpID*, POPayOutID*, PODelID*, POVendID*, PODate, POAmount)
Foreign key POEmpID references TEmployee
Not Null
On Delete Restrict
Foreign key POPayOutID references TPaymentOut
Not Null
On Delete Restrict
Foreign key PODelID references TDelivery
Not Null
On Delete Restrict
Foreign key POVendID references TVendor
Not Null
On Delete Restrict

TPurchaseOrderLine (POLID, POLPOID*, POLTVID*, POLQuantity, POLAmount)
Foreign key POLPOID references TPurchaseOrder
Not Null
On Delete Restrict
Foreign key POLTMID references TTireVendor
Not Null
On Delete Restrict

Differences between ERD and Normalized Relations

An Entity Relationship Diagram (ERD) is a graph-like representation that depicts the relationships between various entities in a database. Normalized Relations show how each table contains only one type of information. For example, one table will provide information about customers' orders while another table will have information on each customer. Entity Relationship Diagrams focus on how different entities are related while Normalize Relations focus on what information is stored in each table.

These differences are important because they ensure that a database is easy to use while being well-organized. Entity Relationship Diagrams assist more so in the phase of planning by showing a broad view of how everything works together. Normalized Relations assist in the phase of implementation by being certain that all of the information is stored efficiently.

Referential Integrity

Referential integrity is a concept pertaining to databases that ensures the relationships between the tables are uniform. The referential integrity constraint ensures that a primary key column of one table matches with a foreign key column of another table. As an example, pertaining to the Sooner Tire Company, we will look at the tables “Customers” and “SalesOrder.” The “Customer” table contains information on each customer including their name, address, contact information, etc. The primary key used in this table is “CustID” which is unique for each customer. The “SalesOrder” table contains information about each sales order including the date, time, etc, and this table contains a foreign key that references the “CustID” from the “Customers” table. This ensures that each sales order has a relationship with a valid customer.

Without a referential integrity constraint, the data would become corrupted. If someone were to remove a customer from the “Customers” table and did not remove the corresponding order from the “SalesOrder” table, the “Orders” table would have values that don’t correspond to a primary key from the “Customers” table. This would, once again, cause the data to be inconsistent leading to incorrect results from queries ran. By using the referential integrity constraint, these potential problems will not appear allowing the data to be as accurate as possible.

Physical Design and Implementation

Physical Design and implementation is the process of transforming the logical database design into a physical database schema that can be implemented into a database management system. This translation involves making decisions about data storage structures, indexing strategies, and breaking up the data. The physical design involves decision making about the storage of data physically and defining indexes. This process also involves factors such as security and performance. After the physical design is finalized, the implementation phase begins in which the database schema is created, the database management system to support the schema is configured, and the data is loaded into the database. This phase tests the database's performance and ensures all requirements are met. Physical Design and Implementation is important to comply with regulations, ensure the database runs smoothly, and keeps the database safe as well as recoverable. We will be using Microsoft SQL Server for the database.

Data Dictionary

A data dictionary is information in a set that includes descriptions of the data elements, metadata, and other various aspects of a database. It is a map for developers to make the data easier to analyze. It gives a thorough description of the data including the format, meaning, relationships between various elements, and the rules used. It can be used as a reference tool for data analysis and modeling as well as it documents the data lineage used in auditing.

Data dictionaries are important because they ensure accuracy and efficiency in the management of data. It describes the data concisely allowing confusion to be eliminated; it also rids the threat of errors in data processing. It allows for good communication among those who work with the data. Examples of a data dictionary can be seen below:

Table	Field Name	PK?	Data Type	Size	Null	References (Foreign Key)	Sample
TCustomer	CustID	Y	Int(Auto Increment)		Not Null		1105-1227
	CFName	N	Varchar	50	Not Null		Joe
	CLName	N	Varchar	50	Not Null		Biden
	CDoorNo	N	Varchar	10	Not Null		1524
	CStName	N	Varchar	50	Not Null		Asp Ave

Denormalization

Denormalization is the process of adding precomputed redundant data to an otherwise normalized relational database to improve the read performance of the database. We demonstrate denormalization in our ERD by getting rid of PaymentInType and payment Outtype to combine then into one table called PaymentType. We also removed the service-type table.

Implemented Physical Design

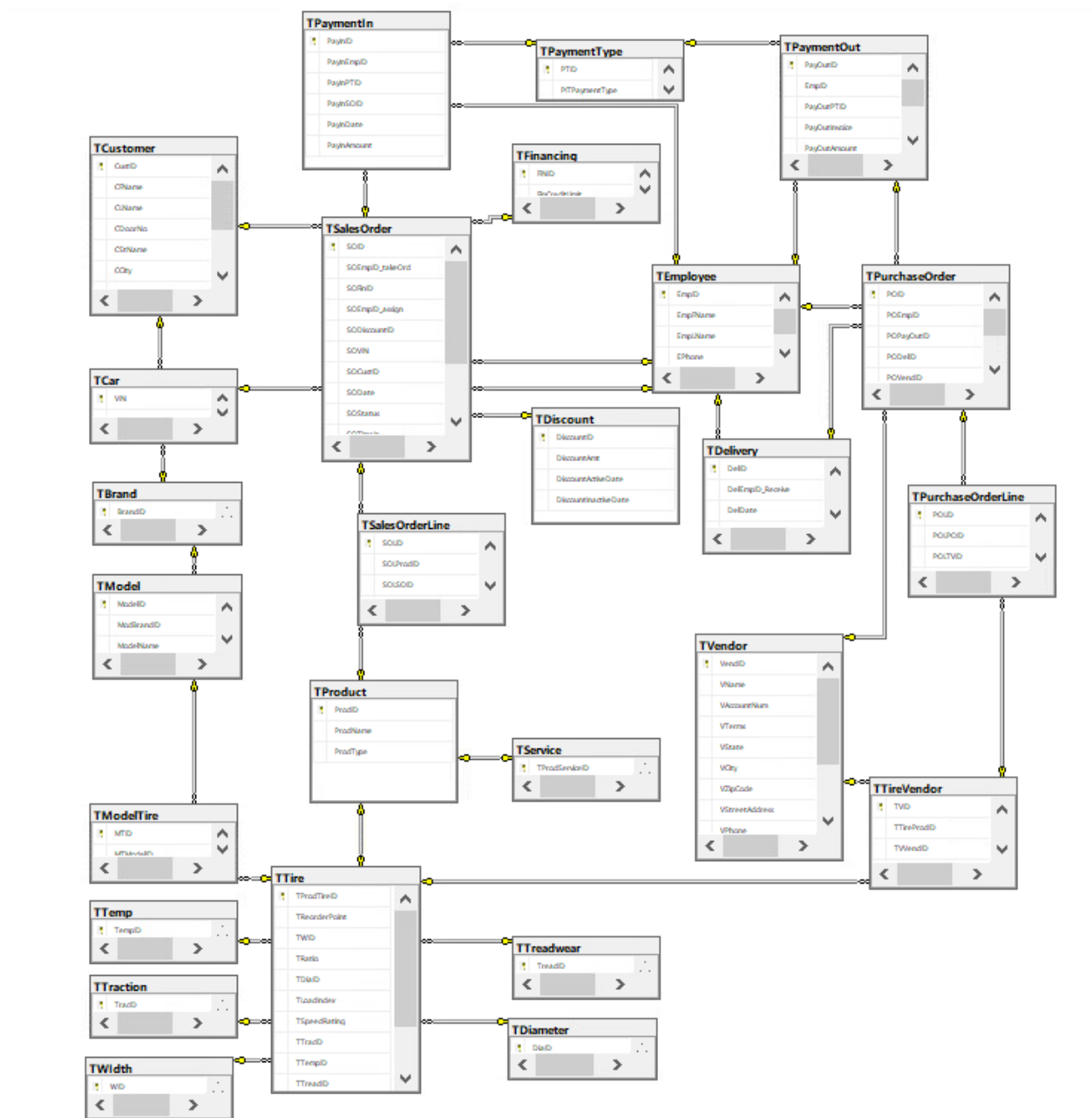


Fig: Database Diagram for our Implemented ERD

Challenges Faced/Addressed During Implementation

Prior to creating the database, we had to ensure the ERD was as strong as possible, so the data could be implemented smoothly. One of the biggest challenges we faced was ensuring everything was added into the code after it was exported to the SQL Server from Lucid Charts. We had to guarantee that the code regarding the primary keys, foreign keys, and any attributes that needed to be labeled “not null” were in place prior to creating the tables by running the code. To overcome this challenge, we carefully navigated each table to find where additional code was needed. Moreover, we used the examples from class and Canvas as the foundation of our code. Overall, the most difficult challenge was managing Walton between five different people. It

required a lot of communication between team members as well as additional group meetings to overcome this.

Strengths and Weaknesses Encountered During Implementation

Our main strength is our communication skills, which allowed us to build our database accurately and efficiently. Using these skills let us stay on top of our due dates. Our team has very compatible skillsets which made us work together fluidly. Our main weakness was the creation of the database, and the timing of it. In the beginning of creating it, we underestimated how long it would take to thoroughly complete it.

Specific SQL Statements Requested

SQL Statements are used in order for data to be stored, manipulated, and retrieved from a database. To create and run a successful statement, information must be looked at and pulled from the ERD created. To solve the problems provided by our client—Sonner Tire Company—and from the created ERD, we created SQL statements that are then put into Walton Labs. If the statements were successfully run, an output in the form of a table was created based on the results.

Q #	Question	SQL
1.	Total sales (in dollars) by state 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)	<pre>Create proc P1SalesByStateForGivenTireVendorAndBrand @vendor varchar(100), @Brand varchar(50) As Begin Select VState, BrandName, VName, sum(SOAmount) TotalSales From (Select * from TBrand B Join TCar Ca on B.BrandID = Ca.CarBrandID Join TSalesOrder SO on ca.VIN=SO.SOVIN Join TSalesOrderLine SOL on SO.SOID=SOL.SOLSOID Join TProduct P on SOL.SOLProdID=P.ProdID) Q1 Join (select * from TVendor V join TTireVendor TV on V.VendID=TV.TVVendID Join TTire T on TV.TTireProdID = T.TProdTireID) Q2 on Q1.ProdID=Q2.TProdTireID Where VName=@vendor and BrandName=@Brand Group by VState, BrandName, VName End</pre> <p>Run this to execute: exec P1SalesByStateForGivenTireVendorAndBrand 'Tire Depot', 'Honda'</p>

Partial Output

	VState	BrandName	VName	TotalSales
1	CA	Honda	Tire Depot	2284.64
2	FL	Honda	Tire Depot	3645.64

- 2 total sales (in dollars) by a customer in a given year
- ```
Create Proc P2CustomerInAYear
@year varchar(4)
As
Begin
select top 5 CustID, CFName, CLName, sum(SOAmount)
as totalAmountSales
from TCustomer C Join TSalesOrder SO
on C.CustID = SO.SOCustID
Where year(SODate)= @year
Group by CustId, cfname, CLName
Order by totalAmountSales desc
End
```

**Run this to execute:**

exec P2CustomerInAYear '2023'

- 3 The five highest selling tires.
- ```
Create Proc P3Top5SellingTiresByQuantity
As
Select Top 5 P.ProdID,P.ProdName, Count(SOLID) as
NoOfQuantitySold
from TSalesOrderLine SOL join TProduct P on
SOL.SOLProdID=P.ProdID
where P.ProdType='Tire'
Group by ProdID,ProdName
Order By NoOfQuantitySold desc
```

Run this to execute:

Exec P3Top5SellingTiresByQuantity

- 4 Itemized invoices for jobs for each customer that need to include tires purchased/tire rotation/tire repair/tire protection.
- ```
Create proc P4invoicebyCustIDandSoDate
@custId int ,
@SoDate date as
Begin
Select custID,CFName, CLname, ProdType, ProdName,
TProdServiceID, SOLAmount, SOLQuantity
From TCustomer C
join TSalesOrder SO
on C.CustID=SO.SOCustID
Join TSalesOrderLine SOL
on SOL.SOLSOID= SO.SOID
Join TProduct P
on SOL.SOLProdID= P.ProdID
join Tservice S
on p.ProdID=S.TProdServiceID
Where CustID=@custId and SO.SODate=@ SoDate
End
```

**Run this to execute:**

exec P4invoicebyCustIDandSoDate 1120,'2023-04-23'

5. The number and type of job performed by each of
- ```
Create View V5TypeOfJobsCompletedPerEmployee
as
SELECT EmpID, EmpFName, EmpLName,
COUNT(EmpID) AS NumberOfJobCompleted,
CASE
WHEN ProdType = 'Tire' THEN 'Sold ' + ProdName
ELSE ProdName
```

	CustID	CFName	CLName	totalAmountSales
1	1150	Brad	Paisley	10000.00
2	1145	Blake	Shelton	9000.00
3	1140	Eric	Church	8000.00
4	1135	Mason	Williams	7000.00
5	1130	Olivia	Gonzalez	6000.00

	ProdID	ProdName	NoOfQuantitySold
1	PROD026	205/65R26	25
2	PROD013	275/65R18	20
3	PROD017	255/70R17	19
4	PROD023	225/60R16	18
5	PROD021	195/65R17	18

	custID	CFName	CLName	ProdType	ProdName	TProdServiceID	SOLAmount	SOLQuantity
1	1120	Emily	Smith	Service	Lifetime Protection	PROD033	67.62	3

	EmpID	EmpFName	EmpLName	NumberOfJobCompleted	TypeOfJobPerformed
1	44337	Emily	Brown	3	Sold 245/60R20
2	44340	William	Brown	3	Sold 265/65R18
3	44340	William	Brown	3	Sold 215/45R17
4	44337	Emily	Brown	2	Free Services - Repair
5	44343	Emma	Nguyen	2	Free Services - Rotate
6	44340	William	Brown	2	Sold 205/65R16

```

our employees.      END AS TypeOfJobPerformed
FROM TSalesOrder SO
JOIN TEmployee E ON SO.SOEmpID_assign = E.EmpID
JOIN TSalesOrderLine SOL ON SOL.SOLSOID =
SO.SOID
JOIN TProduct P ON P.ProdID = SOL.SOLProdID
WHERE SOSStatus = 'closed'
GROUP BY EmpID, EmpFName, EmpLName,
CASE
    WHEN ProdType = 'Tire' THEN 'Sold ' + ProdName
    ELSE ProdName
END
Select *
From V5TypeOfJobsCompletedPerEmployee
Order By NumberOfJobCompleted desc

```

Run this to execute:

```

Select *
FROM V5TypeOfJobsCompletedPerEmployee
Create Proc P6NoOfFreeServicesApplied
as
Select ProdName, Sum(SOLQuantity) as
NumberOfTimesFreeServiceApplied
From TSalesOrderLine SO JOIN TProduct P
on SO.SOLProdID=P.ProdID
Where ProdName like 'Free%'
Group By ProdName

```

Run this to execute:

```
exec P6NoOfFreeServicesApplied
```

	ProdName	NumberOfTimesFreeServiceApplied
1	Free Services - Install	30
2	Free Services - Lifetime Protection	37
3	Free Services - Repair	48
4	Free Services - Rotate	27

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).

7. The following items for Purchase Orders: manufacturer name, number of POs, total cost.

```

Create proc P7TotalCostNumbPOByVendor
AS
Select VName, sum(TVPrice*POLQuantity) as TotalCost,
count(POID) as numberOfPO
From TpurchaseOrder PO Join TPurchaseOrderLine POL
on PO.POID = POL.POLPOID
Join TTireVendor TV
on TV.TVID=POL.POLTVID
Join TVendor V
on V.VendID = TV.TVVendID
Group by Vname

```

Run this to execute:

```
exec P7TotalCostNumbPOByVendor
```

	VName	TotalCost	numberOfPO
1	All-Terrain T/A	3745.00	2
2	BFGoodrich	2107.65	4
3	Big O Tires	1605.00	1
4	Big Tires Inc.	7210.25	5
5	Canadian Tire	2505.00	1
6	Continental	449.90	1
7	Continental AG	1443.75	3
8	Cooper	1735.00	1
9	Cooper Tires	3005.00	2
10	Discount Tire	6356.50	3

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.
- Create Proc
P8NumOrdersAndTotalSalesPerCustInPast2Years as
SELECT C.CustID, count(SOID) as NumOfOrders,
Sum(SOAmount) as TotalAmt
FROM TCustomer C join TSalesOrder SO on C.CustID =
SO.SOCustID
WHERE C.CustID=C.CustID and SO.SODate > '2021-04-23'
GROUP BY C.CustID

	CustID	NumOfOrders	TotalAmt
1	1105	3	3000.00
2	1106	1	2000.00
3	1107	1	3000.00
4	1108	1	4000.00
5	1109	1	5000.00

Run this to execute:

Exec P8NumOrdersAndTotalSalesPerCustInPast2Years

9. List of tires that have not been purchased within the last 6 months (in order to better manage inventory).
- create proc P9TiresNotPurchasedInLast6Months as
SELECT P.ProdName
FROM TTire Tire left join TProduct P on
Tire.TProdTireID=P.ProdID join TSalesOrderLine SOL on
P.ProdID=SOL.SOLProdID join TSalesOrder SO on
SOL.SOLSOID=SO.SOID
WHERE SO.SODate <= '2022-10-23'

	ProdName
1	195/65R17
2	275/65R18
3	215/60R16
4	255/40R19
5	235/50R18

Run this to execute:

Exec P9TiresNotPurchasedInLast6Months

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
- Create Proc
P10CustomerWhoTookFinancingOptionAndRemainingBalance
as
SELECT T1.CFName, T1.CustId, T1.CLName,
F.FinType,T1.Total_Purchase,T1.total_amount_paid,
T1.Remaining_Balance
FROM
(
SELECT SQ1.SOID, SQ1.CustId, SQ1.CFName,
SQ1.CLName, SQ1.SOFinID,
Total_Purchase,total_amount_paid, Total_Purchase-
total_amount_paid as Remaining_Balance
FROM (
SELECT SOID, CustId, C.CFName, CLName,
SOFinID, SUM(SOLAmount *SOLQuantity) as
Total_Purchase
FROM TCustomer C
JOIN TSalesOrder SO on C.CustID=SO.SOCustID
JOIN TSalesOrderLine SOL on SO.SOID =
SOL.SOLSOID
GROUP BY SOID, CustId, C.CFName,
CLName,SOFinID
) SQ1

	CFName	CustId	CLName	FinType	Total_Purchase	total_amount_paid	Remaining_Balance
1	Rosa	1105	Gonzalez	Semiannually	2236.56	610.00	1626.56
2	Jessica	1110	Taylor	Semiannually	2123.77	520.00	1603.77
3	Anthony	1115	Jackson	Semiannually	952.75	720.00	232.75
4	Emily	1120	Smith	Semiannually	1415.90	690.00	725.90
5	John	1125	Garcia	Term Loan	6657.67	450.00	6207.67

than 6 months, all displayed from the latest date and then the largest amount owed.

```

JOIN (
    SELECT S1.SOID, SUM(noOfTimesPaid*
    PayInAmount) total_amount_paid
    FROM (
        SELECT SOID, PayInAmount, COUNT(*) as
        noOfTimesPaid
        FROM TPaymentIn P
        JOIN TSalesOrder S ON P.PayInSOID =
        S.SOID
        GROUP BY SOID, PayInAmount
    ) as T1
    JOIN TSalesOrder S1 ON T1.SOID=S1.SOID
    GROUP BY S1.SOID
) SQ2 ON SQ1.SOID = SQ2.SOID
) as T1 join TFinancing F on T1.SOFinID =F.FINID

```

Run this to execute:

exec P10CustomerWhoTookFinancingOptionAndRemainingBalance

11 Total profit per tire type and manufacturer in the past 6 months.

```

create proc
P11TotalProfitPerTireTypeAndManufacturerLastSixMonths
AS
SELECT BrandName, SOLAmount - TVPrice as Profit,
SODate
FROM TBrand b JOIN TCar c
    ON b.BrandID=c.CarBrandID
    JOIN TSalesOrder so
    ON so.SOVIN=c.VIN
    JOIN TSalesOrderLine sol
    on so.SOID=sol.SOLSOID
    JOIN TProduct p
    ON sol.SOLProdID=p.ProdID
    JOIN TTire ti
    ON p.ProdID=ti.TProdTireID
    JOIN TTireVendor tv
    ON ti.TProdTireID=tv.TTireProdID
WHERE SODate BETWEEN '10-1-2022' and '04-30-
2023'
order by Profit desc

```

Run this to execute:

exec P11TotalProfitPerTireTypeAndManufacturerLastSixMonths

	BrandName	Profit	SODate
1	Chevrolet	938.56	2023-04-23
2	Chevrolet	906.05	2023-04-23
3	Chevrolet	903.05	2023-04-23
4	Chevrolet	903.05	2023-04-23
5	Chevrolet	903.05	2023-04-23
6	Chevrolet	900.80	2023-04-23
7	Chevrolet	900.80	2023-04-23
8	Chevrolet	899.80	2023-04-23
9	Honda	787.63	2023-04-23
10	Honda	786.38	2023-04-23
11	Honda	785.13	2023-04-23
12	Honda	785.13	2023-04-23

- 12 List of all customers that have not made a purchase within the last 12 months from the current date.
- ```
create proc P12CustomerWithNoPurchase
AS
Select distinct custID, CFname, CLName
From TCustomer C left Join TSalesOrder So
on C.custID = SO.SoCustID
Where So.SOCustID is Null
exec P12CustomerWithNoPurchase
```
- Run this to execute:**  
Exec P12CustomerWithNoPurchase
- 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.
- ```
Create proc
P13CustomerWhoseAvergeSalesLessThanAverageAllSale
s
as
select custID, CFName, CLName
from TCustomer C Join TSalesOrder So
On C.custID = SO.SoCustID
group by CustID, CFName, CLName
Having avg(So.SOAmount)<(
Select AVG(SOAmount)
From TSalesOrder
)
```
- Run this to execute:**
exec P13CustomerWhoseAvergeSalesLessThanAverageAllSales

	custID	CFname	CLName
1	1116	Stephanie	Hernandez
2	1117	Christopher	Garcia
3	1118	Mary	Martinez
4	1119	Nicholas	Williams
5	1121	Joshua	Davis
6	1122	Katie	Williams
7	1123	Michael	Brown
8	1124	Sarah	Johnson
9	1126	Rachel	Davis
10	1127	Daniel	Martinez

	custID	CFName	CLName
1	1105	Rosa	Gonzalez
2	1106	Megan	Smith
3	1107	Brian	Davis
4	1108	Ashley	Nguyen
5	1109	Thomas	Lee
6	1110	Jessica	Taylor
7	1115	Anthony	Jackson
8	1120	Emily	Smith
9	1125	John	Garcia

Three Additional Queries

The table below offers queries created by The A Team that we believe Sonner Tires might find useful in the near future. We have created three queries with information as to why we believe they are important, the actual SQL code, a screenshot of some of the output, and a recap of the findings.

Q #	Question	Why is this important	SQL	Partial Output																																													
1	To find the total number of jobs that are left to be done by employee	To track the employee accomplishment	Create view V1NumbJobsLeftEachEmp as Select empID,EmpFName, EmpLName, count(empID) as NumbOFJobsLeft From TEmployee E Join TSalesOrder SO	<table><tr><th></th><th>empID</th><th>EmpFName</th><th>EmpLName</th><th>N</th></tr><tr><td>1</td><td>44335</td><td>Jane</td><td>Smith</td><td>3</td></tr><tr><td>2</td><td>44336</td><td>Michael</td><td>Johnson</td><td>2</td></tr><tr><td>3</td><td>44338</td><td>William</td><td>Taylor</td><td>1</td></tr><tr><td>4</td><td>44339</td><td>Isabella</td><td>Garcia</td><td>1</td></tr><tr><td>5</td><td>44340</td><td>William</td><td>Brown</td><td>1</td></tr><tr><td>6</td><td>44341</td><td>Sophia</td><td>Wilson</td><td>2</td></tr><tr><td>7</td><td>44342</td><td>Ethan</td><td>Kumar</td><td>1</td></tr><tr><td>8</td><td>44344</td><td>Alexander</td><td>Gomez</td><td>1</td></tr></table>		empID	EmpFName	EmpLName	N	1	44335	Jane	Smith	3	2	44336	Michael	Johnson	2	3	44338	William	Taylor	1	4	44339	Isabella	Garcia	1	5	44340	William	Brown	1	6	44341	Sophia	Wilson	2	7	44342	Ethan	Kumar	1	8	44344	Alexander	Gomez	1
	empID	EmpFName	EmpLName	N																																													
1	44335	Jane	Smith	3																																													
2	44336	Michael	Johnson	2																																													
3	44338	William	Taylor	1																																													
4	44339	Isabella	Garcia	1																																													
5	44340	William	Brown	1																																													
6	44341	Sophia	Wilson	2																																													
7	44342	Ethan	Kumar	1																																													
8	44344	Alexander	Gomez	1																																													

- 2 Number of customers by state who made purchase
- To find which state to target for advertisement to increase sales

```

On E.EmpID =
SO.SOEmpID_assign
Where SOStatus in ( 'Open'
,'Pending', 'confirmed')
Group by empFName,
EmpLName,EmpID
Run this to execute:
select * from
V1NumbJobsLeftEachEmp
Create proc
P2ExtraQCustomerByState
as
Select CState, count(CustID) as
totalNumCustomer
From TCustomer C Join TCar
Ca
on C.CustID =
Ca.CarCustID
Join TSalesOrder SO
on SO.SOVIN =
Ca.VIN
Group by CState
Order by totalNumCustomer
desc

```

	CState	totalNumCustomer
1	CO	4
2	OH	4
3	AL	3
4	CA	2
5	TX	2
6	DE	2
7	FL	2
8	NY	2

- 3 Given a brand name we need to find number of tires and services sold for that brand
- To determine what product have we sold and how many to particular brand

```

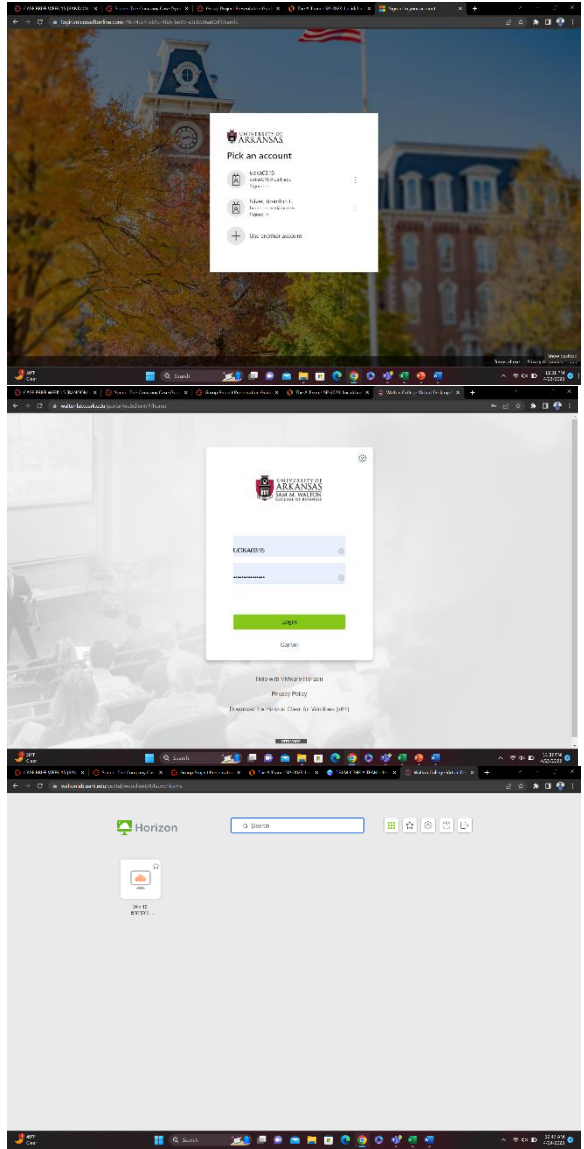
Run this to execute:
exec
P2ExtraQCustomerByState
Create proc
P3ExtraNoProdByBrand
@BrandName varchar(50)
As
Begin
Select Prodtype, count(*) as
NumbOfItemSold
from TBrand B Join TCar C
on B.brandID =
C.CarBrandID
Join TSalesOrder SO
on C.VIN = So.SOVIN
Join TSalesOrderLine
SOL
on SO.SOID =
SOL.SOLSOID
Join TProduct P
On SOL.SOLProdID =
P.ProdID
Where
BrandName=@BrandName
Group by ProdType
End
Run this to execute:
exec P3ExtraNoProdByBrand
'Honda'

```

	Prodtype	NumbOfItemSold
1	Service	23
2	Tire	127

User Documentation

The following table will show how you can access the database and can use the queries that our team has made. It will also show you how to go through the queries and see the results following them.

- | Step # | Image |
|---|---|
| 1. Open a browser and search Walton Lab |  |

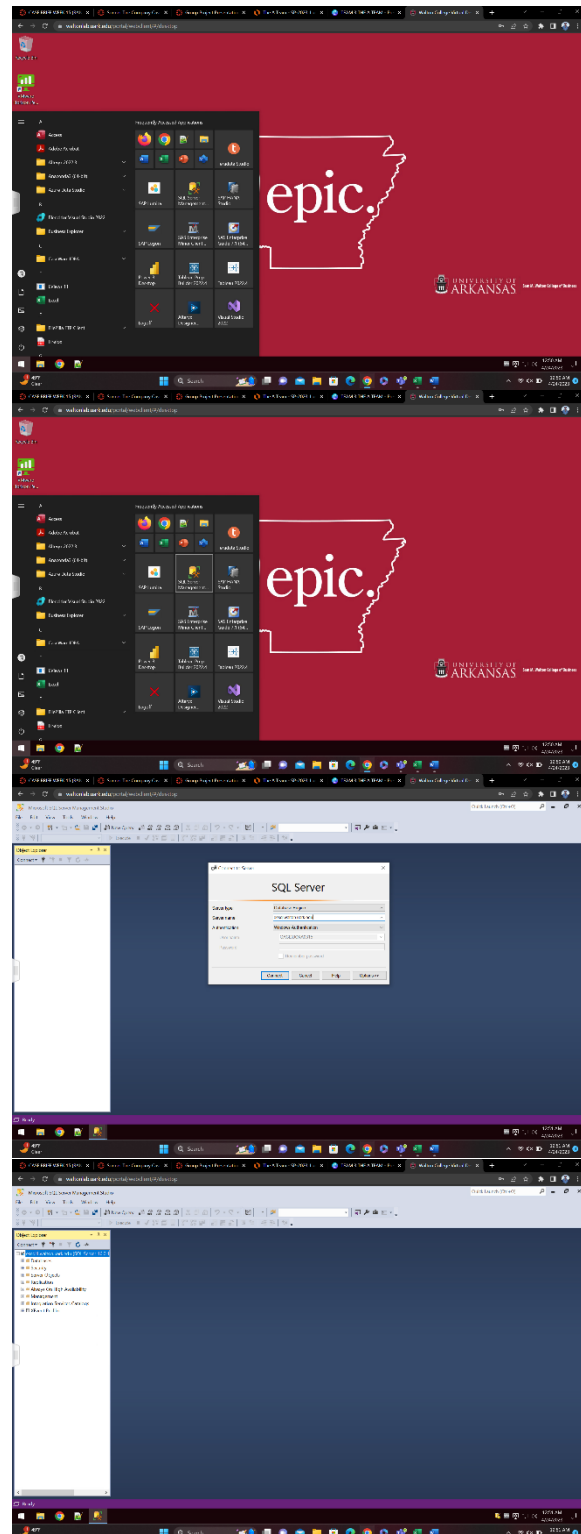
Link:

<https://login.microsoftonline.com/79c742c4-e61c-4fa5-be89-a3cb566a80d1/saml2>

- | |
|--|
| 2. Login to Walton using your personalized username and passcode |
|--|

- | |
|------------------------------|
| 3. Click on the “Win 10” box |
|------------------------------|

4. Click the windows logo on the bottom left corner of the computer
5. Select the “SQL Server Management Studio” option (found in the first row of big, box icons)
6. You will be prompted with a SQL Server screen, keep all the information the same but in the “Server” box enter the following line:
Essql1.walton.uark.edu
7. View the panel that popped up on the left-hand side of the screen and locate the “Databases” folder

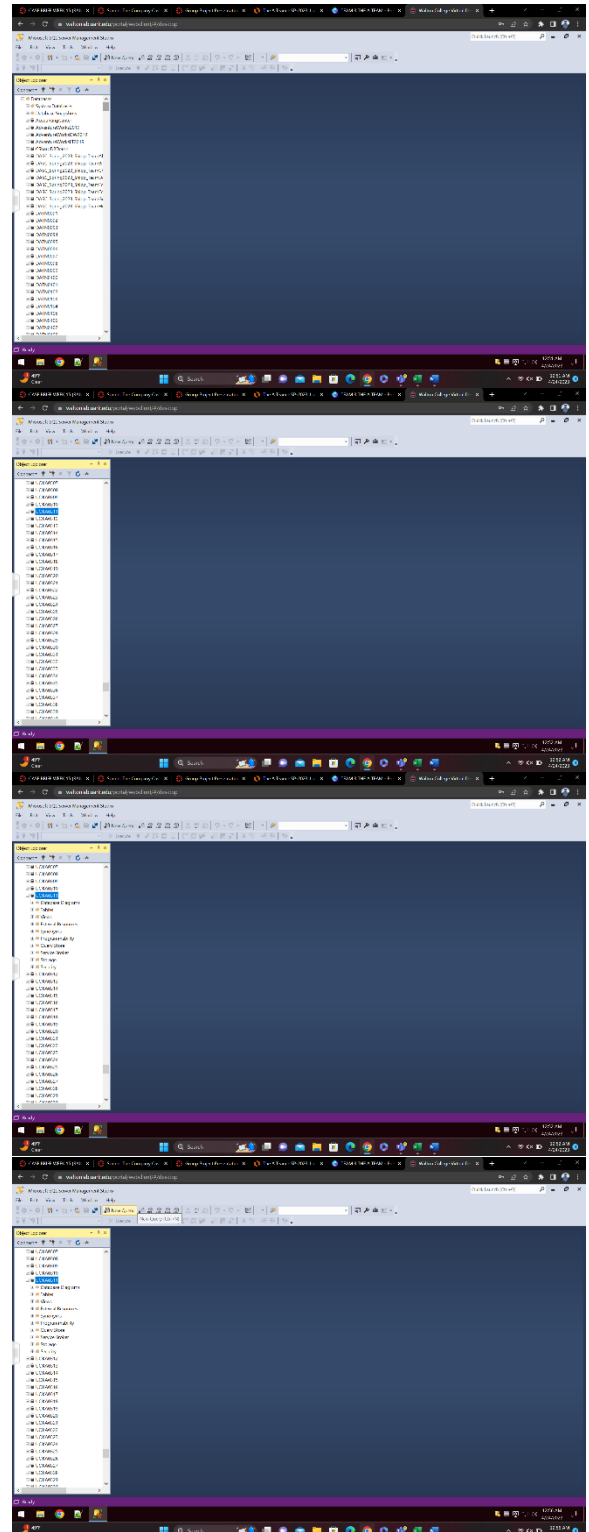


9. Locate the folder named “UOKA0311”

9. Locate the folder named “UOKA0311”

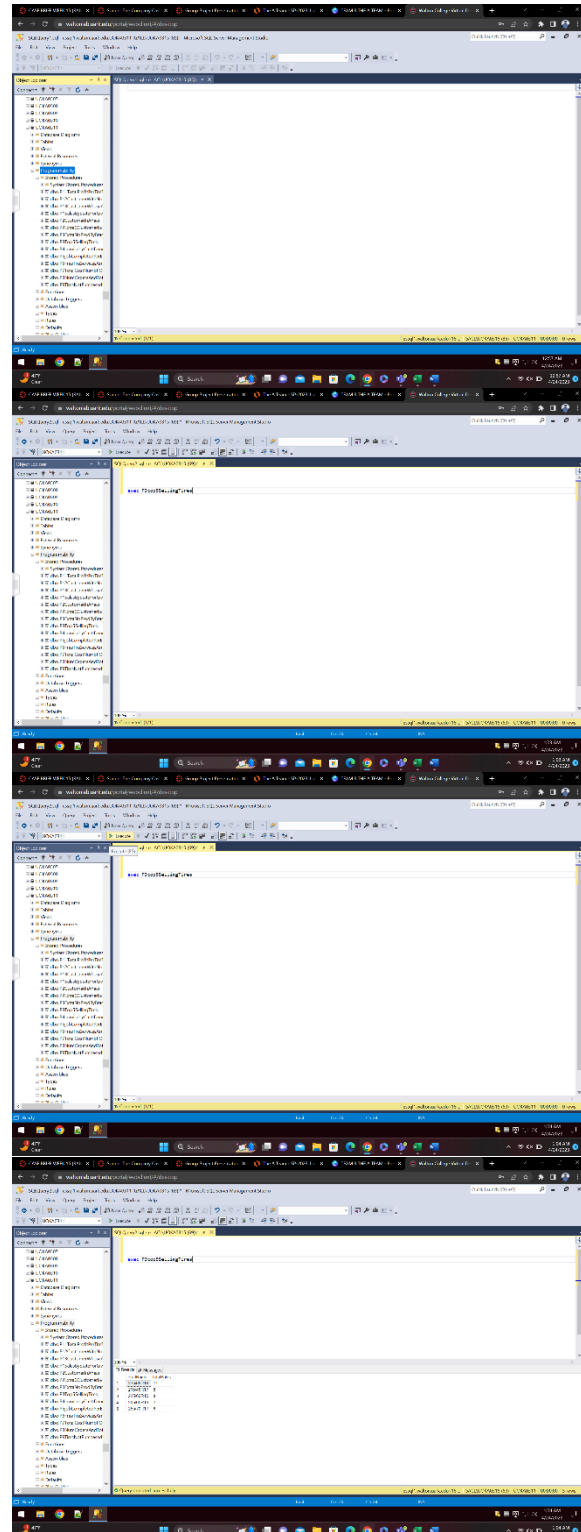
11. Locate the “Programmability” folder

11. Locate the “Programmability” folder



12. Drop down the “Programmability” folder, then drop down the “Stored Procedures” folder to see the list of procedures available to use (for the example, we will use P3top5SellingTires)
13. Click on the white space and type “exec {name of whatever procedure you choose}”
14. Click the Execute button found on the tool bar
15. Look below the white space and you will see a table has been generated and information has been displayed

Note: If the stored procedure requires a parameter/s please provide. Look at what it is expecting in the above Query section. Also, for Views, instead of programmability, click on the Views Folder and choose the View that you want to run. You can run Views like a normal sql query.



Lessons Learned

As a team we learned how to work together to complete separate sections of a large project. While we all showed success in different areas, we worked both separately and as a whole to achieve the best results. Our team worked incredibly well together, and we often found ourselves having too many volunteers to complete each task. With high motivation, the A Team worked incredibly hard to create this document. In addition, every team member upheld the duties that were asked of them, which were laid out in the team contract at the beginning of the semester.

Member Name: What you learned:

Benjamin Finley	Over the course of the project, I've gained a much deeper understanding of ERDs and the complexity that comes with them. One more thing that I've learned through this process is the process of logical design and how specific and precise your ERD must be to work well. Lastly, I've improved my ability to work in a team setting.
Brandon Niver	Throughout this project I have learned several things. One of the most prominent things that I learned is the overall understanding of our evolving ERD. Another thing that I learned is how to create a functioning logical design as well as how to operate a collaborative work effort on Walton Lab. In addition, I learned how to trust my teammates to complete vital tasks of a project instead of doing all of the sections myself, I believe this lesson will help me immensely in the near future.
Roshan Rai	I learnt how to use databases by adding data, removing data, and adding tables. I learned the whole process of how the things fit together from the first day of class till today. The whole thing together from scratch. The important skill I learned is how to troubleshoot the problems in the database. Many times, there were all kinds of errors in the database, Excel and our team had to find so many ways to solve it as each problem required a different solution.
Meg Marrs	Throughout this project I have learned how to put trust in my team to finish work on time and correctly. Because of this I had the ability to focus on what I needed to get done for the project instead of worrying about others. I had the ability to follow along with our ERD and learn much better how they work in a database.
Presley Lindsay	I learned how to collaboratively work on a group project efficiently while dividing the work up in order for individuals and the group to be successful. I also learned how much an ERD can evolve, and overall, how assumptions varying from group to group can lead the entire project to be different. I learned how to work as a group on Walton Lab through heightened communication and planning skills.

Team Function/Insights

As a team, we learned a lot throughout this project. We worked together to complete separate sections of a large project, while we all showed success in different areas, we worked both separately and to achieve the best results.

One of the most important things we learned was the value of communication and trust in a team. By dividing up the work and trusting each other to complete our assigned tasks, we were able to work efficiently and effectively. We also learned the importance of planning and staying organized, as our evolving ERD required constant updates and modifications.

Individually, each team member gained valuable skills throughout the project. For example, some team members gained a deeper understanding of ERDs and logical design, while others learned more about database troubleshooting and collaboration. But what was most important was that we all upheld the duties that were asked of us, which were laid out in our team contract at the beginning of the project.

Overall, we are incredibly proud of the work we accomplished as a team. We faced challenges and overcame them together, and we learned a lot about the power of teamwork and communication in a collaborative project.

Appendix

Team Contract

Listed below is an image of the team contract that we created prior to the start of this project. This contract includes the team's name, logo, moto, all the team members contact information, time and

days available to meet, expectations for the peer evaluation, behavior that will constitute point deductions, and presentation date preferences.



Team Name: The A-Team

Logo:

Team Motto: Teamwork makes the dream work

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Brandon Niver	brandonniver@ou.edu brandonniver@gmail.com	918-527-3226	Organization, communication	Anytime after 6 everyday
Ben Finley	Benjamin.n.finley-1@ou.edu Ben.finley19@gmail.com	469-400-8528	Time management, adaptability/flexibility	Any time after 4 on Tuesday/Thursday/Friday Not Monday/Wednesday
Roshan Rai	Roshan.raai@ou.edu Roshkiratrai@gmail.com	405-408-5349	Good at SQL	Monday: 9-11 am Tuesday/Wednesday: after 6 Friday all day (subject to change)
Presley Lindsay	Presley.a.lindsay-1@ou.edu presleyalindsay@yahoo.com	832-788-7202	Good at preparing presentations, good at SQL	Free in the morning and anytime after 6
Margaret Marrs	Margaret.o.marrs_1@ou.edu	720-582-1585	Organization, creativity	Anytime Friday Tuesday/Thursday anytime except 10:30-4:30

Team Expectations for the confidential peer evaluation :

All members fulfill their assigned jobs, good communication to one another, and doing work on time and completely.

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

Giving work to teammates or submitting work late, being disrespectful, and not meeting group standards for quality of work.

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

April 10, April 19, March 22

Data Dictionary Model

The Data Dictionary is a sheet that provides a user with the description of the attributes, tables, and records of data in a database. It provides a reference for users to better understand the structure and format of the data to for more accurate and precise use. It can contain the table

name, attribute name, primary and foreign keys, the data type and size, and samples of records for users to reference.

Table	Field Name	PK?	Data Type	Size	Null	References (Foreign Key)	Sample
TCustomer	CustID	Y	Int(Auto Increment)		Not Null		1105-1227
	CFName	N	Varchar	50	Not Null		Joe
	CLName	N	Varchar	50	Not Null		Biden
	CDoorNo	N	Varchar	10	Not Null		1524
	CStName	N	Varchar	50	Not Null		Asp Ave
	CCity	N	Varchar	50	Not Null		Norman
	CState	N	Char	2	Not Null		OK
	CZip	N	Varchar	5	Not Null		73019
	CPhone	N	Varchar	12	Not Null		405-325-3611
	CEmail	N	Varchar	64	Not Null		swetha@ou.edu
TDiscout	DiscountID	Y	Int(Auto Increment)		Not Null		10-36
	DiscountAmt	N	Int		Not Null		100
	DiscountActiveDate	N	Date		Not Null		1/1/2023
	DiscountInactiveDate	N	Date		Not Null		1/1/2025
TWidth	WID	Y	Int		Not Null		8000-8008
	Width	N	Int		Not Null		195
TDiameter	DialID	Y	Int		Not Null		500-519
	DiaMeas	N	Int		Not Null		625
TTemp	TempID	Y	Int		Not Null		400-402
	TempGrade	N	Varchar	1	Not Null		A
TTraction	TracID	Y	Int		Not Null		800-803
	TracGrade	N	Varchar	2	Not Null		AA
TTreadwear	TreadID	Y	Int				900-906
	TreadGrade	N	Int				90
TProduct	ProductID	Y	Varchar	8	Not Null		PROD005- PROD037
	ProdName	N	Varchar	100	Not Null		245/45R19
	ProdType	N	Varchar	50	Not Null		Tire
TTire	TireProdID	N	Int(Auto Increment)		Not Null	TProduct	PROD010
	TWID	N	Int		Not Null	TWidth	195
	TDiaMeas	N	Int		Not Null	TDiaMeas	625
	TTempGrade	N	Int		Not Null	TTempGrade	A
	TTracID	N	Int		Not Null	TTracID	AA
	TTreadID	N	Int		Not Null	TTreadID	90
	TReorderPoint	N	Int		Not Null		16
	TQTY_OH	N	Int		Not Null		80

TService	ServiceProdID	Y	Varchar	50	Not Null	TProduct	PROD030
	ServiceDes	N	Varchar	1000	Not Null		Tire Rotation
TBrand	BrandID	Y	Int(Auto Increment)		Not Null		1-5
	BrandName	N	Varchar	40	Not Null		Honda
TModel	ModelID	Y	Int(Auto Increment)		Not Null		2000-2105
	ModBrandID	N	Int		Not Null	TBrand	5
	ModName	N	Varchar	255	Not Null		Civic
	ModYear	N	Varchar	10	Not Null		2023
TCar	VIN	Y	Varchar		Not Null		1C3CCCAB6FN674604 (Random)
	CarBrandID	N	Int		Not Null	TBrand	5
	CarCustID	N	Int		Not Null	TCustomer	1105
TModelTire	TMID	Y	Int(Auto Increment)		Not Null		40000-40332
	TMModelID	N	Int		Not Null	TModel	2000
	TMProdID	N	Int		Not Null	TProduct	PROD005
TEmployee	EmpID	Y	Int(Auto Increment)		Not Null		44334-44435
	EFName	N	Varchar	50	Not Null		Swetha
	ELName	N	Varchar	50	Not Null		Siripurapu
	EPhone	N	Varchar	12	Not Null		405-325-3611
	EEmail	N	Varchar	50	Not Null		swetha@ou.edu
TDelivery	DelID	Y	Int(Auto Increment)		Not Null		99101-99210
	DelEmpID-Receive	N	Int		Not Null	TEmployee	44334
	DelDate	N	Date		Null		1/1/2023
	DelTrackingInfo	N	Varchar	100	Not Null		874-569-3214
	DelCarrierInfo	N	Varchar	100	Not Null		USPS
TFinancing	FINID	Y	Int(Auto Increment)		Not Null		10000-10006
	FinType	N	Varchar	50			Term Loan
TSalesOrder	SOID	Y	Int(Auto Increment)		Not Null		7600-7701
	SOCustID	N	Int		Not Null	TCustomer	1105
	SOEmpID_TakeOrd	N	Int		Null	TEmployee	44334
	SOEmpID_Assign	N	Int		Null	TEmployee	44334
	SODiscountID	N	Int		Null	TDiscout	11
	SOVIN	N	Int		Not Null	TCar	1GNEK13Z23J105418
	SODate	N	Date		Not Null		1/1/2023
	SOSatus	N	Varchar	50	Not Null		"Finished"
	SOTimeIn	N	Time		Not Null		12:00:00

	SOTimeOut	N	Time		Not Null		12:00:00
	SOInsurance	N	Bit		Not Null		Y/N
TSalesOrderLine	SOLID	Y	Int(Auto Increment)		Not Null		78118-78593
	SOLProdID	N	Int		Not Null	TProduct	PROD005
	SOLSOLID	N	Int		Not Null	TSalesOrder	7609
	SOLQuantity	N	Int		Not Null	TDelivery	34
	SOLStatus	N	Varchar	50	Not Null		Finished
TPaymentType	PTID	Y	Int(Auto Increment)		Not Null		1000-1004
	PTPaymentType	N	Varchar	50	Not Null		Cheque
TPaymentIn	PayInID	Y	Int(Auto Increment)		Not Null		777100-777390
	PayInEmpID	N	Int		Not Null	TEmployee	44334
	PayInPTID	N	Int		Not Null	TPaymentType	1000
	PayInSOLID	N	Int		Not Null	TSalesOrder	7600
	PayInDate	N	Date		Not Null		1/1/2023
	PayInAmount	N	Decimal	(8,2)	Not Null		262.89
	PayInStatus	N	Varchar	50	Not Null		Complete
TPaymentOut	PayOutID	Y	Int(Auto Increment)		Not Null		666108-666216
	PayOutEmpID	N	Int		Null	TEmployee	44334
	PayOutPTID	N	Int		Not Null	TPaymentType	1000
	PayoutInvoice	N	Int	250	Not Null		7890
	PayOutDate	N	Date		Not Null		1/1/2023
	PayOutAmount	N	Decimal	(8,2)	Not Null		5369.42
	PayOutStatus	N	Varchar	50	Not Null		Complete
TVendor	VendID	Y	Int(Auto Increment)		Not Null		452-581
	VName	N	Varchar	50	Not Null		Michelin
	VAccountNum	N	Int		Not Null		TK123
	VTerms	N	Varchar	50	Not Null		Net 60
	VStAddress	N	Varchar	50	Not Null		307 W Brooks St
	VCity	N	Varchar	15	Not Null		Norman
	VState	N	Char	2	Not Null		OK
	VPhone	N	Varchar	12	Not Null		405-325-3611
	VContactFName	N	Varchar	50	Not Null		Swetha
	VContactLName	N	Varchar	50	Not Null		Siripurapu
	VBulkDiscount	N	Bit		Not Null		0/1
TTireVendor	TVID	Y	Int(Auto Increment)		Not Null		97500-97706
	TVProdID	N	Int		Not Null	TProduct	PROD005

	TVVendID	N	Int		Not Null	TVendor	452
	TVPrice	N	Decimal	(10, 2)	Not Null		15632.12
	TVMileage	N	Int		Not Null		60000
TPurchaseOrder	POID	Y	Int(Auto Increment)		Not Null		11320-11438
	POEmpID	N	Int		Not Null	TEmployee	44334
	POPayOutID	N	Int		Not Null	TPaymentOut	666108
	PODelID	N	Int		Not Null	TDelivery	99209
	POVendID	N	Int		Null	TVendor	521
	PODate	N	Date		Not Null		1/1/2023
TPurchaseOrderLine	POLID	Y	Int(Auto Increment)		Not Null		556-860
	POLPOID	N	Int		Not Null	TPurchaseOrder	11320
	POLTVID	N	Int		Not Null	TTireVendor	97500
	POLQuantity	N	Int		Not Null		100

Project Management

Project Start Date	3/6/2023			Project End Date	4/30/2023
	Student Name	Duration (Min)	% Complete	Subtotal Minutes	Subtotal Cost
Milestone 1					
Read Case + Prepare Questions for client	Ben Finley	45	100	45	\$19

Read Case + Prepare Questions for client	Brandon Niver	45	100	45	\$19
Read Case + Prepare Questions for client	Roshan Rai	45	100	45	\$19
Read Case + Prepare Questions for client	Meg Marrs	45	100	45	\$19
Read Case + Prepare Questions for client	Presley Lindsay	30	100	30	\$13
Client Meeting	Ben Finley	10	100	10	\$4
Client Meeting	Brandon Niver	10	100	10	\$4
Client Meeting	Roshan Rai	10	100	10	\$4
Client Meeting	Meg Marrs	10	100	10	\$4
Client Meeting	Presley Lindsay	10	100	10	\$4
ERD Design	Ben Finley	45	100	45	\$19
ERD Design	Brandon Niver	45	100	45	\$19
ERD Design	Roshan Rai	45	100	45	\$19
ERD Design	Meg Marrs	45	100	45	\$19
ERD Design	Presley Lindsay	45	100	45	\$19
Assumptions	Ben Finley	30	100	30	\$13
Assumptions	Brandon Niver	15	100	15	\$6
Assumptions	Roshan Rai	15	100	15	\$6
Assumptions	Meg Marrs	15	100	15	\$6
Assumptions	Presley Lindsay	15	100	15	\$6
Write-up preparation	Ben Finley	60	100	60	\$25
Write-up preparation	Brandon Niver	60	100	60	\$25
Write-up preparation	Roshan Rai	60	100	60	\$25
Write-up preparation	Meg Marrs	60	100	60	\$25
Write-up preparation	Presley Lindsay	60	100	60	\$25
Sub Total		875		875	\$365
Milestone 2					
ERD Revision	Ben Finley	30	100	30	\$13
ERD Revision	Brandon Niver	30	100	30	\$13
ERD Revision	Roshan Rai	120	100	120	\$50
ERD Revision	Meg Marrs	30	100	30	\$13
ERD Revision	Presley Lindsay	30	100	30	\$13
Logical Design	Ben Finley	90	100	90	\$38
Logical Design	Brandon Niver	90	100	90	\$38
Logical Design	Roshan Rai	90	100	90	\$38
Logical Design	Meg Marrs	120	100	120	\$50
Logical Design	Presley Lindsay	90	100	90	\$38
Write-up Revision	Ben Finley	60	100	60	\$25
Write-up Revision	Brandon Niver	60	100	60	\$25

Write-up Revision	Roshan Rai	60	100	60	\$25
Write-up Revision	Meg Marrs	60	100	60	\$25
Write-up Revision	Presley Lindsay	60	100	60	\$25
Sub Total		1020		1020	\$425
Milestone 3					
ERD Revision	Ben Finley	45	100	45	\$19
ERD Revision	Brandon Niver	45	100	45	\$19
ERD Revision	Roshan Rai	60	100	60	\$25
ERD Revision	Meg Marrs	30	100	30	\$13
ERD Revision	Presley Lindsay	100	100	100	
Data Dictionary	Ben Finley	75	100	75	\$31
Data Dictionary	Brandon Niver	45	100	45	\$19
Data Dictionary	Roshan Rai	50	100	50	\$21
Data Dictionary	Meg Marrs	30	100	30	\$13
Data Dictionary	Presley Lindsay	100	100	100	
Physical Design Work	Ben Finley	45	100	45	\$19
Physical Design Work	Brandon Niver	60	100	60	\$25
Physical Design Work	Roshan Rai	120	100	120	\$50
Physical Design Work	Meg Marrs	60	100	60	\$25
Physical Design Work	Presley Lindsay	100	100	100	
SQL Work	Ben Finley	60	100	60	\$25
SQL Work	Brandon Niver	60	100	60	\$25
SQL Work	Roshan Rai	120	100	120	\$50
SQL Work	Meg Marrs	60	100	60	\$15
SQL Work	Presley Lindsay	100	100	100	
Write-up Revision	Ben Finley	45	100	45	\$19
Write-up Revision	Brandon Niver	60	100	60	\$25
Write-up Revision	Roshan Rai	30	100	30	\$13
Write-up Revision	Meg Marrs	60	100	60	\$15
Write-up Revision	Presley Lindsay	100	100	\$0	
Sub Total				665	\$307
Final Submission					
SQL Work	Ben Finley	45	100	45	\$19
SQL Work	Brandon Niver	45	100	45	\$19
SQL Work	Roshan Rai	120	100	120	\$50
SQL Work	Meg Marrs	45	100	45	\$19
SQL Work	Presley Lindsay	65	100	65	\$27
Write-up Revision	Ben Finley	45	100	45	\$19
Write-up Revision	Brandon Niver	45	100	45	\$19
Write-up Revision	Roshan Rai	45	100	45	\$19
Write-up Revision	Meg Marrs	45	100	45	\$19

Write-up Revision	Presley Lindsay	45	100	45	\$19
Sub Total				255	\$106
			Total	2920	\$1,217