Packwood Ski Company

Golden Retrievers - Final Submission

Brandon Witte, Sam Corley, Trevor Doerr, Sophie Graham

MIS 3353 - Database Management

Professor Jeremy Bellah

December 3, 2018

Executive Summary

1. PROJECT

The primary focus of Golden Retrievers was to assist Packwood Ski Company (PSC) as they outgrew their current data processes and adopted a new database system which unified records and reporting. Through this experience, Golden Retrievers aimed to get clarifications from the client and assess their needs. Once information was collected, our goal became organizing and implementing this information so that it would benefit the client's needs; to do this, we structured our data within an Entity Relationship Diagram (ERD), completed the logical design for the database, and implemented the database in SQL Server. With the intent of providing value to PSC's customers and increasing PSC's operational efficiency, Golden Retrievers designed reports for PSC to understand the datas meaning and help their representatives make quality decisions for their company. While we hoped to provide optimal results of PSC, we worked to follow our budget and stay on schedule as we developed the database.

2. OBJECTIVES

The main goal of this project was to create a more efficient way for PSC to collect, store, and use raw data in a way that was helpful to their productivity. Our objective for this system was to make it easy for PSC to understand and run it themselves. We attacked this by including our physical ERD as well as conceptual and logical designs that further delved into the specifics in each table, attribute and relation.

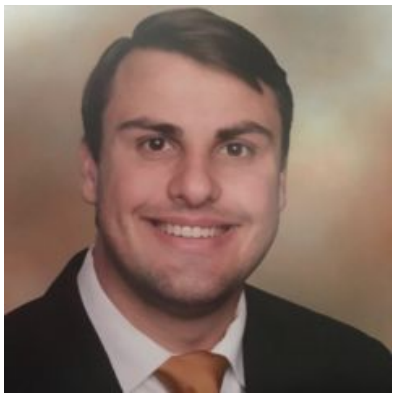
3. MAIN RESULTS

The database produced tangible results of things like total sales by region in a given month and number of distinct products managed by each product manager. These along with many other different combinations of data will help PSC streamline their activities to recognize where there is profit and where there is loss.

Contents

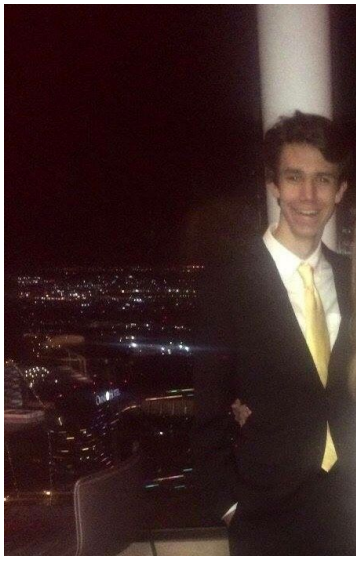
Get to Know the Team: Golden Retrievers

Brandon Witte



Major: Management Information Systems
Year: Senior
Brief Background:
Born in Plano, Texas. Passionate in business and technology which led me to pursue a career in MIS. Currently taking two upper division MIS courses and graduating December, 2019. Fun Fact: Can make some really good steak and potatoes.

Sam Corley



Major: Management Information Systems and Sports Management
Year: Senior
Brief Background:
Born in Edmond, Oklahoma. Passionate about sports and the statistics behind them which led me pursuing a major in both MIS and Sports management. My dream job is to work alongside the Thunder basketball recruiting team.

Sophie Graham

Major: Management Information Systems; Entrepreneurship and Venture Management
Minor: Finance
Year: Senior
Brief Background:
Born in Oklahoma City, Oklahoma. Driven to succeed and eager to embrace new challenges, Sophie works to better herself and help others improve their knowledge and leadership through her everyday responsibilities and goals.

Trevor Doerr

Major: International Business; Management Information Systems
Year: Senior
Brief Background:
Born in Denver, Colorado. Spent last summer studying in Berlin, Germany and the summer before that interning at a construction management software start-up called Buildertrend. Interested in working in the corporate side of musical instrument manufacturing. Love music, sports and anything outdoors.

Conceptual Design

The conceptual phase is the early phase of the design process which involves gathering research and modeling it into an appropriate outline.

The Client Meeting

In the client meeting, our team discussed past problems that Packwood Ski Company has had in the past, as well as how they would like their data to be organized in the future. With the goal of setting a clear outline of the company's business model, we asked specific questions about production, customers and sales, and raw materials ordering.

- Meeting Time: Monday, October 15th
- Location: Adams Hall Rm 4
- Interviewers: Brandon Witte, Sophie Graham, Sam Corley, and Trevor Doerr
- Interviewee: Dr. Sandberg

Q&A During the Meeting & Information We Learned

Products

- What data issues did the company have in the past?
    - Excel spreadsheets. Don't know which versions are correct.
- Do we need to keep track of items in the bundle bindings/special production runs?
    - No.
- Are components manufactured stored on site or off site?
    - On site
- What kind of info do you want to record regarding supplies of raw material?
    - How much is on hand
- Can a product manager manage multiple product categories?
    - Can manage multiple
- Do we need to keep any information regarding Packwood manufacturing facility?
    - no
- What does a product manager do and what kind of information is relevant for the company to have about them?
    - General info about them
- Would you like to track which skis come pre assembled?(ski cores)
    - Don't need to keep track of different types

Customer and Sales

- What info do you want to keep track of the customers' sales rep?
    - Which employee is assigned to customer
    - One employee per region; one region can have many customers
- Can a customer have multiple sales reps?
    - No
- If the salesperson hand delivers, are they separate from employees that ship to customers? Is this salesperson the sales rep for our customers?
    - Different shipping method. Could be, but not necessarily - could be warehouse manager
- Can the salesperson hand-deliver multiple packages?
    - Any shipment can have multiple sales orders.
    - Each shipment can only be made to one customer
- Does customer size = order size?
    - Customer size can be measured by which customers are purchasing large amounts
    - Which customers are large customers in general?
    - We want to categorize customers by how large they are as an organization and pull that info to target the right customers.
- What type of information do you need to record on your balance forward and invoice? (ItemID, ItemQuantity, ItemCost, InvoiceValue, InvoiceDate, InvoiceTerms))... Same for Balance Forward?
    - Total Amount (whatever is outstanding at the end of each month)
    - If someone has terms on Balance Fwd, it'll clear their books once a month.
    - Outstanding payment of over 90 days is too far out
    - Could apply
- One order can have one discount?
    - Can have more than one discount.
- Would you like to track when the inventory on a certain model reaches a certain low point?
    - Quantity on hand
    - Reorder Point

Raw Materials Ordering

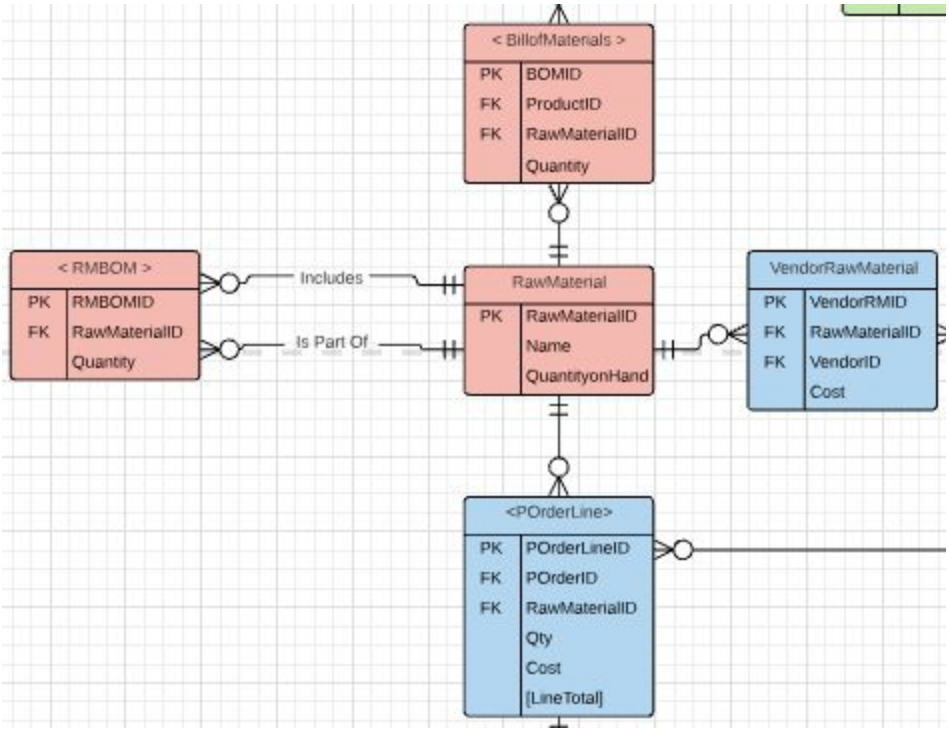
- Do you send a bid for raw materials since it is based on the price offered? Or do you just ask different vendors?
    - Price Offered is on Purchase Order Line
    - Purchase Req is not needed in our ERD's
- Do we need to keep track of any departments?(Such as Marketing or Purchasing)
    - No

- Are manufacturing facilities separate from shops?
    - Yes

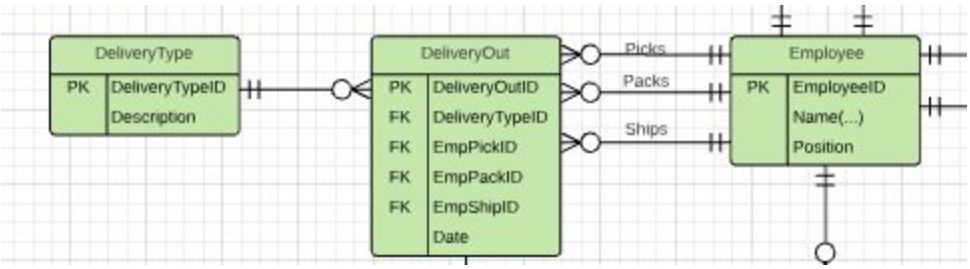
Significant Assumptions

To understand the business of PSC, Golden Retrievers modeled PSC's business information within a database. While there were specific components of the business covered through the client meeting or the reading packet, the Golden Retrievers team made a few assumptions about how PSC's processes are performed and the relationships between their data. Listed below are five significant assumptions made when creating the ERD model:

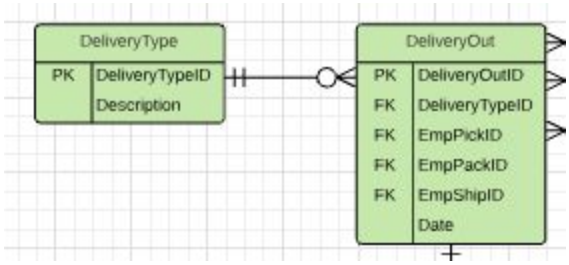
1. Raw materials may come completed or be part of another raw material



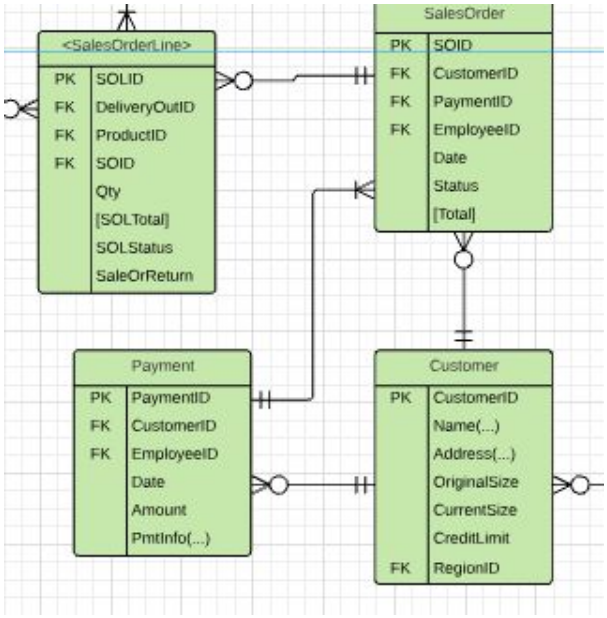
2. PSC wants to keep track of each employee that picked, packed, and shipped an order.
    a. Showed the three relationships between the Employee and Delivery Out entities



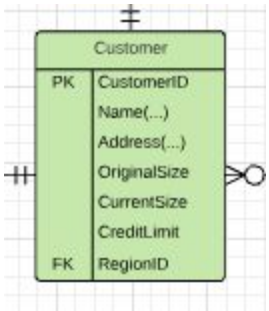
3. PSC wants to keep track of delivery type
    a. Made a reference table for delivery type



4. PSC wants to keep track of the customers payment info
    a. Led us to create the Payment entity in the ERD



5. There is a credit limit for each customer.
    a. Added a attribute “CreditLimit” to the Customer entity



What is an ERD? Why is it necessary?

An ERD, or Entity Relationship Diagram, is a graphical representation of information that depicts relationships between people, objects, things, etc. It is a data modeling technique which can help define business processes and further explain the relationships people and products have with the business, both internally and externally. They provide a visual starting point for database design to help determine requirements for an organization. For example, we show where the information is stored for customers as well as what happens after they purchase a product.

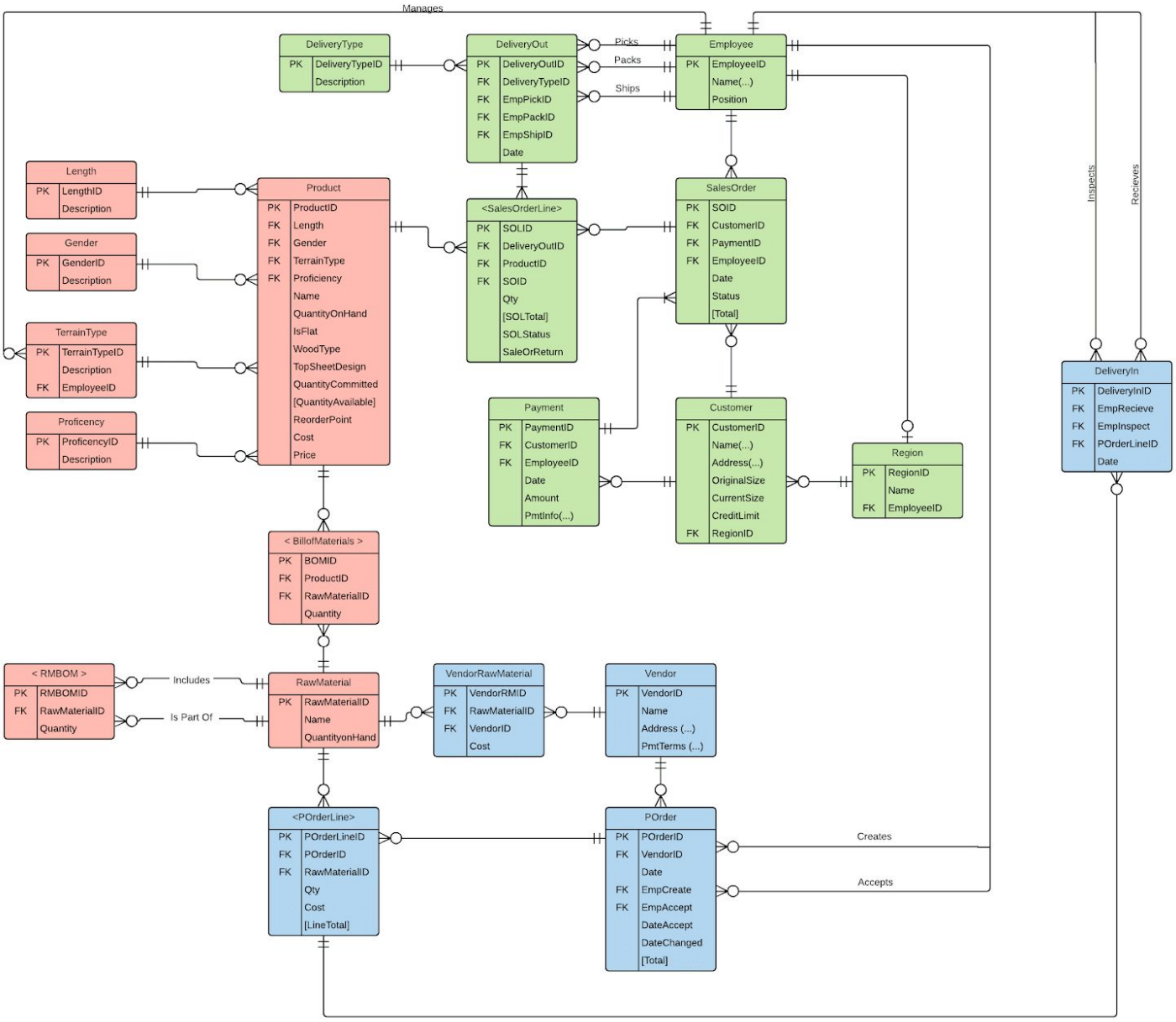
Business Cycles Used

We used three different business cycles in this ERD. We started with the production cycle, which records product and all of its different segments as well as the amount of a specific product that is committed to a sale, available for sale, its cost, price, and when to reorder materials for it. The production cycle keeps track of what raw materials are being used in the products and the quantity on hand of those materials; this includes individual materials that go into making a more complete raw material. The primary activities that take place during the production cycle is product design, planning, operations, and cost accounting.

Connected to this portion is the expenditure cycle. This keeps track of the purchase orders and the approved vendors that you use. The purpose of the expenditure cycle is to ensure that an organization has the supplies, materials, and services it needs to meet its company goals. The three core components of the expenditure cycle include ordering, receiving, and paying invoices.

Finally, we used the revenue cycle to to record all things relating to the selling of products. Here is where information is kept to track customers and employees and how they each fit into the elements of sale and delivery of the product. The core components of the revenue cycle include sales order, pick, pack, and ship, and billing.

ERD Created


Manages
DeliveryType
PK DeliveryTypeID
Description
DeliveryOut
PK DeliveryOutID
FK DeliveryTypeID
FK EmpPickID
FK EmpPackID
FK EmpShipID
Date
Picks
Packs
Ships
Employee
PK EmployeeID
Name(...)
Position
Inspects
Recieves
Length
PK LengthID
Description
Gender
PK GenderID
Description
TerrainType
PK TerrainTypeID
Description
FK EmployeeID
Proficency
PK ProficencyID
Description
Product
PK ProductID
FK Length
FK Gender
FK TerrainType
FK Proficiency
Name
QuantityOnHand
IsFlat
WoodType
TopSheetDesign
QuantityCommitted
[QuantityAvailable]
ReorderPoint
Cost
Price
<SalesOrderLine>
PK SOLID
FK DeliveryOutID
FK ProductID
FK SOID
Qty
[SOLTotal]
SOLStatus
SaleOrReturn
SalesOrder
PK SOID
FK CustomerID
FK PaymentID
FK EmployeeID
Date
Status
[Total]
DeliveryIn
PK DeliveryInID
FK EmpRecieve
FK EmpInspect
FK POrderLineID
Date
Payment
PK PaymentID
FK CustomerID
FK EmployeeID
Date
Amount
PmtInfo(...)
Customer
PK CustomerID
Name(...)
Address(...)
OriginalSize
CurrentSize
CreditLimit
FK RegionID
Region
PK RegionID
Name
FK EmployeeID
< BillofMaterials >
PK BOMID
FK ProductID
FK RawMaterialID
Quantity
< RMBOM >
PK RMBOMID
FK RawMaterialID
Quantity
Includes
Is Part Of
RawMaterial
PK RawMaterialID
Name
QuantityonHand
VendorRawMaterial
PK VendorRMID
FK RawMaterialID
FK VendorID
Cost
Vendor
PK VendorID
Name
Address (...)
PmtTerms (...)
<POrderLine>
PK POrderLineID
FK POrderID
FK RawMaterialID
Qty
Cost
[LineTotal]
POrder
PK POrderID
FK VendorID
Date
FK EmpCreate
FK EmpAccept
DateAccept
DateChanged
[Total]
Creates
Accepts

Changes made to generic ERDs

To fit the business model of Packwood Ski Company, Golden Retrievers has added new entities; these entities reflect the information PSC wants to keep track of within their database.

Revenue Cycle: In our Revenue cycle we added a DeliveryType entity so we could describe if it was a hand delivery or a shipment. On our sales order entity we added a status attribute so the customer could check the status of their order. We also added a SalesAndReturns attribute to track when a product was returned.

Expenditure Cycle: In our Expenditure Cycle we added a VendorRM attribute to track which vendor sold us which raw material. We added a DateRec attribute to track when the product was received from the vendor. We also added a Employee Rec to track which employee signed for the order.

Production Cycle: In our Production Cycle we added category entities differentiate our Products. In our bill of materials entity we added a TopSheetDesign attribute in order to identify which top sheet design was put on the ski. We also added EmployeeID to our Terrain type attribute to track which product manager was in charge of which terrain type.

Logical Design

Logical design is the process of arranging data into a series of logical schema using tables and attributes. Think of logical design as a blueprint for a house that shows everything that needs to be accomplished and how to accomplish it. It transitions the project from something the client can understand to something that can be implemented. Furthermore, logical design makes it where you can fix bad databases as well as it separates the content from the structure to enable building of an efficient, accurate database.

Normalization

Normalization is the way to ensure that your database is trustworthy and efficient. This ensures that a database is easy to navigate and catch any mistakes that were made in the making of the ERD. Normalization is important in this project because it helped give a more detailed version of our multivalued attributes such as Customer Name. Instead of the attribute being only Name, it would separate the attribute into first and last name. This makes it easier to differentiate the attributes and quickly find what a user searches for.

Normalized Relations

Production Cycle

TProduct (ProductID, *Length, *Gender, *TerrainType, *Proficiency, Name, QtyOnHand, IsFlat, WoodType, TopSheetDesign, QtyCommitted, ReorderPoint, Cost, Price)
Foreign key Length references TLength Not Null ON DELETE RESTRICT
Foreign key Gender references TGender Not Null ON DELETE RESTRICT
Foreign key TerrainType references TTerrainType Not Null ON DELETE RESTRICT
Foreign key Proficiency references TProficiency Not Null ON DELETE RESTRICT

TLength (LengthID, Description)

TGender (GenderID, Description)

TTerrainType (TerrainTypeID, *EmployeeID, Description)
Foreign key EmployeeID references TEmployee Not Null ON DELETE RESTRICT

TProficiency (ProficiencyID, Description)

TBillOfMaterials (BOMID, *ProductID, *RawMaterialID, Quantity)
Foreign key ProductID references TProduct Not Null ON DELETE RESTRICT
Foreign key RawMaterialID references TRawMaterial Not Null ON DELETE RESTRICT

TRawMaterial (RawMaterialID, Name, QtyOnHand)

TRMBOM (RMBOMID, *RawMaterialID, Quantity)
Foreign key RawMaterialID references TRawMaterialNot Null ON DELETE RESTRICT

Expenditure Cycle

TVendorRawMaterial (VendorRMID, *RawMaterialID, VendorID, Cost)
Foreign key RawMaterialID references TRawMaterial Not Null ON DELETE RESTRICT
Foreign key VendorID references TVendor Not Null ON DELETE RESTRICT

TVendor (VendorID, Name, Address, Street, City, State)

TPOrderLine (POrderLineID, *POrderID, *RawMaterialID, Qty, Cost)
Foreign key POrderID references TPOrder Not Null ON DELETE RESTRICT
Foreign key RawMaterialID references TRawMaterial Not Null ON DELETE RESTRICT

TPOrder (POrderID, *VendorID, *EmpCreate, *EmpAccept, Date, DateAccept, DateChanged)
Foreign key VendorID references TVendor Not Null ON DELETE RESTRICT
Foreign key EmpCreate references TEmployee Not Null ON DELETE RESTRICT
Foreign key EmpAccepted references TEmployee Not Null ON DELETE RESTRICT

TDeliveryIn (DeliveryInID, *EmpReceive, *EmpInspect, Date)
Foreign key EmpReceive references TEmployee Not Null ON DELETE RESTRICT
Foreign key EmpInspect references TEmployee Not Null ON DELETE RESTRICT

Revenue Cycle

TEmployee (EmployeeID, FName, LName, Position)

TSalesOrder (SOID, *CustomerID, *PaymentID, *EmployeeID, Date, Status)
Foreign key CustomerID references TCustomer Not Null ON DELETE RESTRICT
Foreign key PaymentID references TPayment Not Null ON DELETE RESTRICT
Foreign key EmployeeID references TEmployee Not Null ON DELETE RESTRICT

TCustomer (CustomerID, *RegionID, FName, LName, Address, Street, City, State, OriginalSize, CurrentSize)
Foreign key RegionID references TRegion ON DELETE SET NULL

TPayment (PaymentID, *CustomerID, *EmployeeID, Date, Amount)
Foreign key CustomerID references TCustomer Not Null ON DELETE RESTRICT
Foreign key EmployeeID references TEmployee Not Null ON DELETE RESTRICT

TRegion (RegionID, *EmployeeID, Name)
Foreign key EmployeeID references TEmployee Not Null ON DELETE RESTRICT

TSalesOrderLine (SOLID, *DeliveryOutID, *ProductID, *SOID, Qty, SOLStatus, SaleOrReturn)
Foreign key DeliveryOutID references TDeliveryOut Not Null ON DELETE RESTRICT
Foreign key ProductID references TProduct Not Null ON DELETE RESTRICT
Foreign key SOID references TSalesOrder Not Null ON DELETE RESTRICT

TDeliveryOut (DeliveryOutID, *DeliveryTypeID, *EmpPickID, *EmpPackID, *EmpShipID, Date)
Foreign key DeliveryTypeID references TDeliveryType Not Null ON DELETE RESTRICT
Foreign key EmpPickID references TEmployee Not Null ON DELETE RESTRICT
Foreign key EmpPackID references TEmployee Not Null ON DELETE RESTRICT
Foreign key EmpShipID references TEmployee Not Null ON DELETE RESTRICT

TDeliveryType (DeliveryTypeID, Description)

Differences between ERD and Normalized Relations

ERDs are useful for at least three audiences: database analysts, clients, and database developers. An ERD, or Entity Relationship Diagram, provides a visual outline of data and their relationships to one another; following the information listed within the entities, attributes, and relationships, users understand and reference how data is stored within a database. Normalized Relations focus on three main components: ensuring the field values cannot be broken down further (atomicity), eliminating data duplication problems, and making data well-structured. It is beneficial for creating new data and also remodeling prior databases. As opposed to ERDs, Normalized Relations does not include derived attributes within relations and requires that multi-value attributes be broken down.

Referential Integrity

A referential integrity constraint is used when the relationship between two entities requires a foreign key. Thus, if there is a relationship must match a valid primary key. This is needed because it shows that certain data cannot be entered in the database because it violate the integrity by making some data redundant. For example, null data is sometimes not allowed depending on the relationship between two entities. It also is necessary because it shows why and how the foreign keys are connected and how it will be implemented by the “On Delete Restrict/Set Null/Cascade” and “Null Allowed or Not Null” parts of the constraints. Restrict refers to a one to one relationship, Set Null refers to a zero to many relationship, and Cascade refers to a one to one weak entity relationship. Therefore, having the constraints allows for a better understanding of the database and ensures referential integrity and an efficient database.

Physical Design and Implementation

Physical design, or a “platform-specific” methodology, identifies the best way to integrate the database design into a particular relational database management system (RDBMS); the goal is to develop a database that works quickly.

Data Dictionary

Table	Field Name	Data Type	Size	Null	References
TEmployee					
	EmployeeID	int identity	1,1	NOT NULL	PRIMARY KEY
	EmployeeFName	varchar	200		
	EmployeeLName	varchar	200		
	EmployeePosition	varchar	200		
Tlength					
	LengthID	int		NOT NULL	PRIMARY KEY
	Length	int		NOT NULL	
Tgender					
	GenderID	int		NOT NULL	PRIMARY KEY
	Description	varchar	50	NOT NULL	
TProficiency					
	ProficiencyID	int		NOT NULL	PRIMARY KEY
	Description	varchar	50	NOT NULL	
TterrainType					
	TerrainTypeID	int		NOT NULL	PRIMARY KEY
	Description	varchar	50	NOT NULL	
	EmployeeID	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
Tproduct					
	ProductID	int		NOT NULL	PRIMARY KEY
	LengthID	int		NOT NULL	FOREIGN KEY REFERENCES Tlength (LengthID)
	GenderID	int		NOT NULL	FOREIGN KEY REFERENCES TGender (GenderID)
	TerrainTypeID	int		NOT NULL	FOREIGN KEY REFERENCES TTerrainType (TerrainTypeID)
	ProficiencyID	int		NOT NULL	FOREIGN KEY REFERENCES TProficiency (ProficiencyID)
	Name	varchar	50	NOT NULL	
	QtyOnHande	int		NOT NULL	
	IsFlat	char	1	NOT NULL	
	WoodType	varchar	50	NOT NULL	
	TopSheetDesign	varchar	100	NOT NULL	
	QtyCommitted	int		NOT NULL	
	QtyAvailable	int			
	ReorderPoint	int		NOT NULL	
	Cost	smallmoney		NOT NULL	
	Price	smallmoney		NOT NULL	
TRawMaterial					
	RawMaterialID	int		NOT NULL	PRIMARY KEY
	Name	varchar	100	NOT NULL	
	QtyOnHand	int		NOT NULL	
TRMBOM					
	RMBOMID	int		NOT NULL	PRIMARY KEY
	RawMaterialID	int		NOT NULL	FOREIGN KEY REFERENCES TRawMaterial (RawMaterialID)
	Quantity	int		NOT NULL	
TBOM					
	BOMID	int identity	(1,1)	NOT NULL	PRIMARY KEY
	ProductID	int		NOT NULL	FOREIGN KEY REFERENCES TProduct (ProductID)
	RawMaterialID	int		NOT NULL	FOREIGN KEY REFERENCES TRawMaterial (RawMaterialID)
	Quantity	int		NOT NULL	
Tvendor					
	VendorID	int		NOT NULL	PRIMARY KEY
	VendorName	varchar	200	NOT NULL	
	Address	varchar	500	NOT NULL	
	Street	varchar	(200),	NOT NULL	
	City	varchar	(200),	NOT NULL	
	State	varchar	200	NOT NULL	
TVendorRM					
	VendorRMID	int		NOT NULL	PRIMARY KEY
	RawMaterialID	int		NOT NULL	FOREIGN KEY REFERENCES TRawMaterial (RawMaterialID)
	VendorID	int		NOT NULL	FOREIGN KEY REFERENCES TVendor (VendorID)
	Cost	money		NOT NULL	

TPOrder					
	POrderID	int		NOT NULL	PRIMARY KEY
	VendorID	int		NOT NULL	FOREIGN KEY REFERENCES TVendor (VendorID)
	EmpCreate	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	EmpAccept	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	Date	date		NOT NULL	
	DateAccept	date		NOT NULL	
	DateChange	date		NOT NULL	
TPOrderLine					
	POrderLineID	int		NOT NULL	PRIMARY KEY
	POrderID	int		NOT NULL	FOREIGN KEY REFERENCES TPOrder (TPOrderID)
	RawMaterialID	int		NOT NULL	FOREIGN KEY REFERENCES TRawMaterial (RawMaterialID)
	Qty	int		NOT NULL	
	Cost	money		NOT NULL	
	LineTotal	money			
TDeliveryIn					
	DeliveryInID	int		NOT NULL	PRIMARY KEY
	EmpReceive	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	EmpInspect	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	POrderLineID	int		NOT NULL	FOREIGN KEY REFERENCES TPOrderLine (POrderLineID)
	Date	date		NOT NULL	
Tregion					
	RegionID	int		NOT NULL	PRIMARY KEY
	EmpID	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	Name	text		NOT NULL	
TCustomerID					
	CustomerID	int		NOT NULL	PRIMARY KEY
	RegionID	int		NOT NULL	FOREIGN KEY REFERENCES TRegion (RegionID)
	CustName	varchar	(200),	NOT NULL	
	CustAddress	varchar	500	NOT NULL	
	OriginalSize	int		NOT NULL	
	CurrentSize	int		NOT NULL	

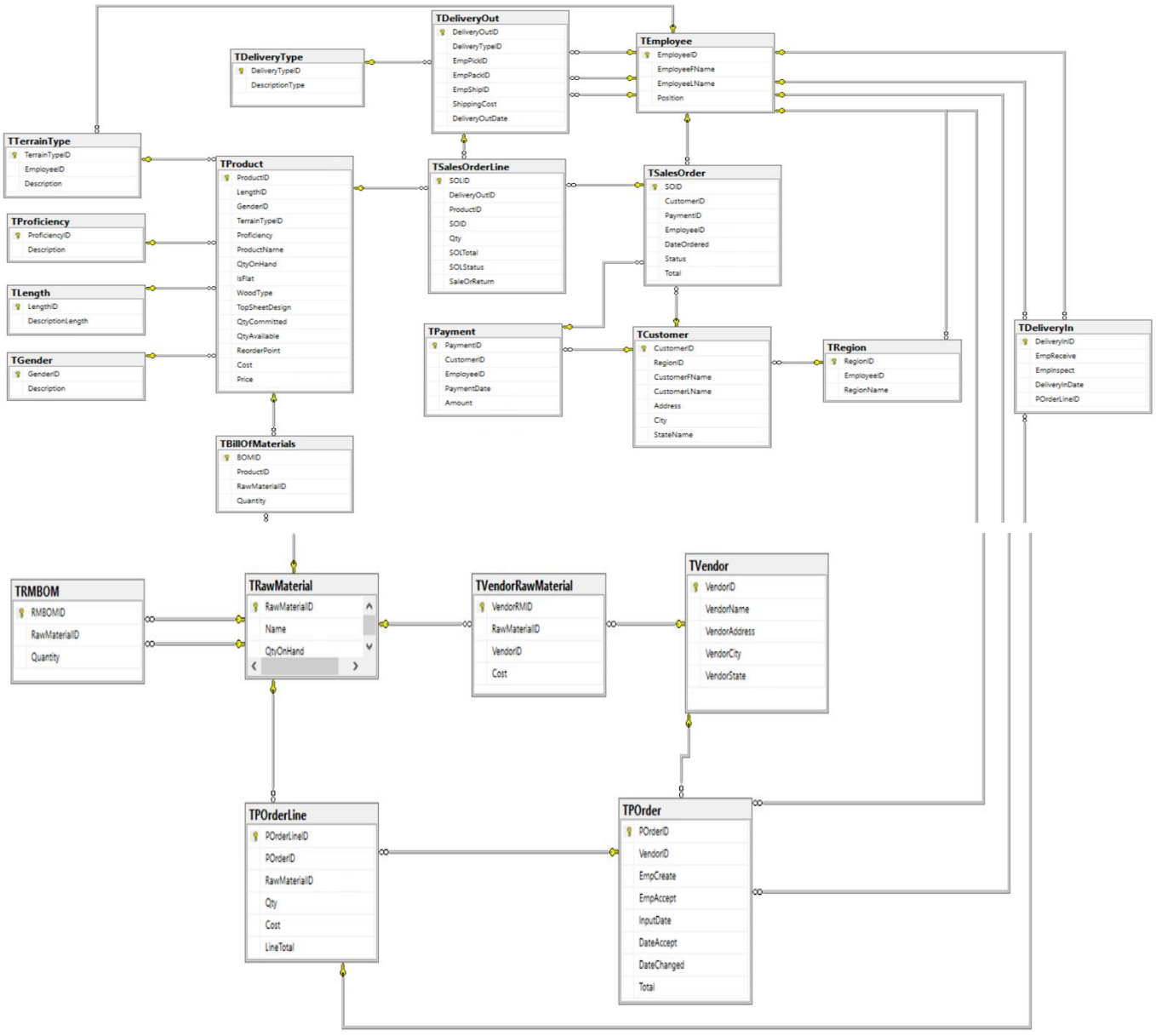
Tpayment					
	PaymentID	int		NOT NULL	PRIMARY KEY
	CustomerID	int		NOT NULL	FOREIGN KEY REFERENCES TCustomer (CustomerID)
	EmpID	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	Date	date		NOT NULL	
	Amount	money		NOT NULL	
	PmtInfo	varchar		NULL	
TSalesOrder					
	SOID	int		NOT NULL	PRIMARY KEY
	CustomerID	int		NOT NULL	FOREIGN KEY REFERENCES TCustomer (CustomerID)
	PaymentID	int		NOT NULL	FOREIGN KEY REFERENCES TPayment (PaymentID)
	EmpID	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	Date	date		NOT NULL	
	Status	text		NOT NULL	
	Total	money		NOT NULL	
TDeliveryType					
	DeliveryTypeID	int		NOT NULL	PRIMARY KEY
	Description	varchar	200	NOT NULL	
TDeliveryOut					
	DeliveryOutID	int		NOT NULL	PRIMARY KEY
	DeliveryTypeID	int		NOT NULL	FOREIGN KEY REFERENCES
	EmpPick	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	EmpPack	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	EmpShip	int		NOT NULL	FOREIGN KEY REFERENCES Temployee (EmployeeID)
	Date	date		NOT NULL	
TSalesOrderLine					
	SOLID	int		NOT NULL	PRIMARY KEY
	DeliveryOutID	int		NOT NULL	FOREIGN KEY REFERENCES TDeliveryOut (DeliveryOutID)
	ProductID	int		NOT NULL	FOREIGN KEY REFERENCES TProduct (ProductID)
	SOID	int		NOT NULL	FOREIGN KEY REFERENCES TSalesOrder (SOID)
	Qty	int		NOT NULL	

	SOLTotal	money		Not NULL	
	SOLStatus	varchar	200	NOT NULL	
	SaleOrReturn	int		NOT NULL	

Denormalization

Denormalization is an approach to speeding up data retrieval performance in a relational database, where the database administrator selectively adds specific instances of redundant data *after* the data structure has been normalized. We chose not to denormalize any of our relationships due to accuracy and database integrity.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

Two of our biggest challenges:

1) Asking significant questions during preparation period

   To better understand what was expected of our team, we met to discuss and write down which questions we had, which areas we could improve on, and how we could better prepare for the upcoming milestone submissions.

2) Underestimating time commitment

   We established weekly times when our team could meet to discuss our objectives and responsibilities. In this way, we organized our time more efficiently.

Strengths and Weaknesses Encountered During Implementation

Strengths:

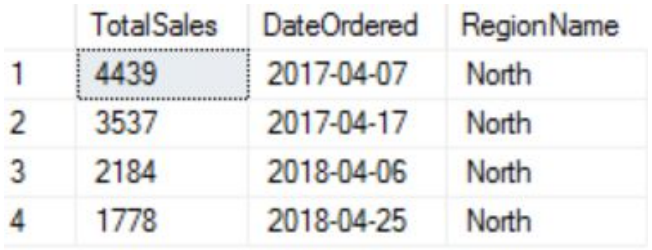
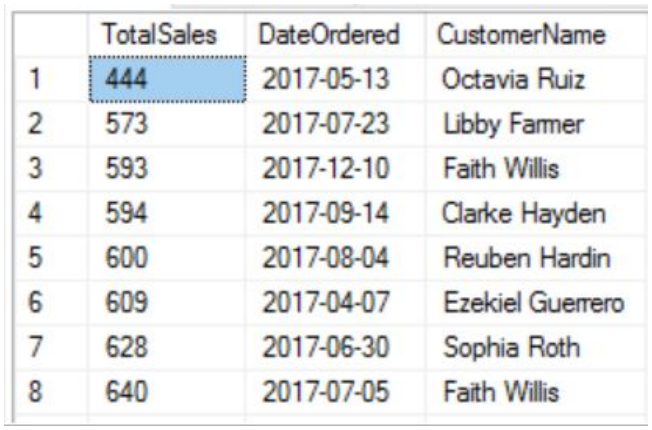
1) Working as a team and making time to meet outside of the classroom setting.
2) Asking questions to the professor and receiving constructive feedback from him and classmates.
3) Being aware of deadlines and ensuring we were organized with our time.

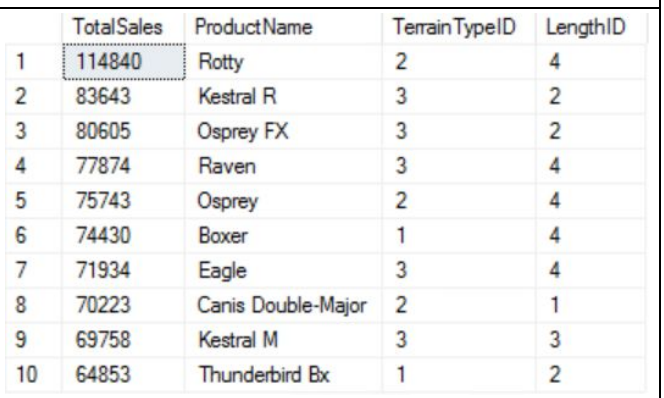
Weaknesses:

1) Keeping our terminology uniform across all users.
2) Understanding how to use SQL server.
3) Creating data without repeating any information.

Specific SQL Statements Requested

Here we explain the specific programs we were asked to execute by the client in the database, as well as additional queries we believe would be useful for the operation. We have included the request asked of us, the SQL code needed to implement the program, and an image of the result of the program.

Query #	Question	SQL	Partial Output
1	What are the total sales (in dollars) by region in a given month?	SELECT Sum(Total) as TotalSales, DateOrdered, RegionName FROM TSalesOrder SO JOIN TCustomer C ON SO.CustomerID = C.CustomerID JOIN TRegion R ON C.RegionID = R.RegionID WHERE Month(DateOrdered) = 04 and RegionName = 'North' GROUP BY RegionName, DateOrdered	
2	What are the total sales (in dollars) by customer in a given year?	select distinct sum(SO.Total) as TotalSales, DateOrdered, (CustomerFName + ' ' + CustomerLName) as CustomerName from TCustomer C join TSalesOrder SO on C.CustomerID = SO.CustomerID Where year(SO.DateOrdered) = 2017	

		group by SO.DateOrdered, CustomerFName, CustomerLName	
3	What is the total expenditure by vendor in a given year?	select distinct(VendorName), sum(Cost) as TotalCost , InputDate From TPOrder PO join TVendor V on PO.VendorID = V.VendorID join TVendorRawMaterial VR on V.VendorID = VR.VendorID where year(InputDate) = 2018 group by InputDate, VendorName	
4	What are the ten highest selling (a) models, (b) terrain ski types, and (c) model -sizes?	SELECT top 10 Sum(Total) as TotalSales, ProductName, TerrainTypeID, LengthID FROM TSalesOrder SO JOIN TSalesOrderLine SOL ON SO.SOID = SOL.SOID JOIN TProduct P ON P.ProductID = SOL.ProductID GROUP BY ProductName, TerrainTypeID, LengthID	

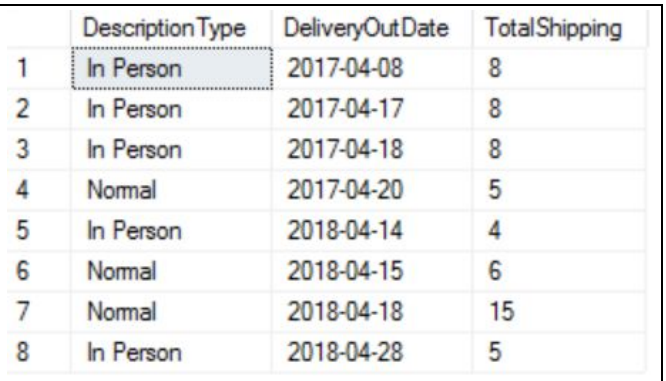
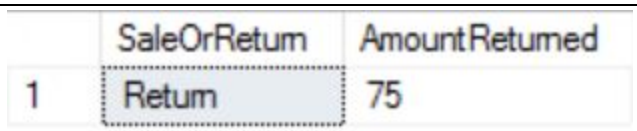
	VendorName	TotalCost	InputDate
1	A Facilisis PC	5	2018-10-19
2	A Scelerisque Sed Limited	789	2018-03-21
3	Adipiscing Ligula Aenean Inc.	2066	2018-01-03
4	Adipiscing Ligula Aenean Inc.	2066	2018-12-21
5	Aenean Gravida PC	513	2018-08-14
6	Amet Ornare Corporation	1495	2018-06-14
7	Ante Dictum Mi Corporation	813	2018-07-16
8	Auctor Velit Consulting	661	2018-02-02

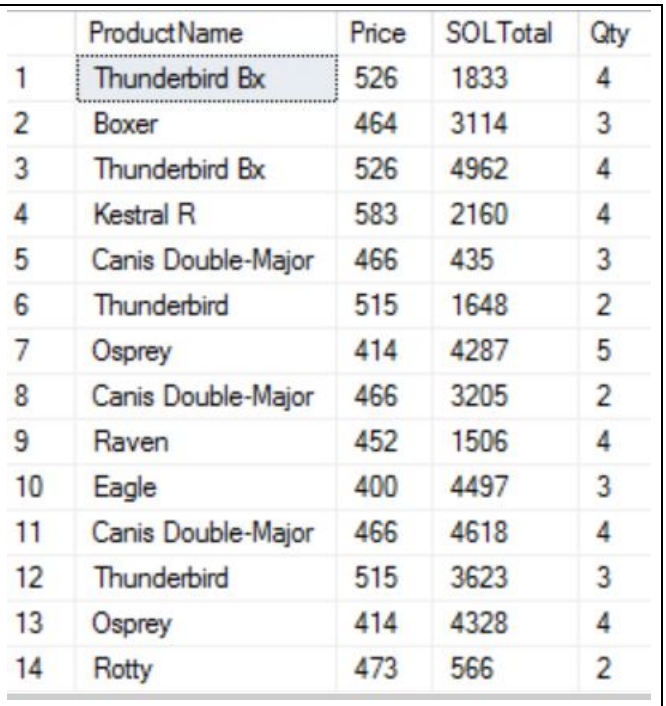
	TotalSales	ProductName	TerrainTypeID	LengthID
1	114840	Rotty	2	4
2	83643	Kestral R	3	2
3	80605	Osprey FX	3	2
4	77874	Raven	3	4
5	75743	Osprey	2	4
6	74430	Boxer	1	4
7	71934	Eagle	3	4
8	70223	Canis Double-Major	2	1
9	69758	Kestral M	3	3
10	64853	Thunderbird Bx	1	2

		ORDER BY TotalSales DESC	
5	What are the ten most profitable products in a given year?	select distinct top 10 P.ProductName, sum(Qty*Price)as TotalProfits, DateOrdered from TSalesOrderLine SOL join TProduct P on P.ProductID = SOL.ProductID join TSalesOrder SO on SOL.SOID = SO.SOID where year(SO.DateOrdered) = 2017 Group by P.ProductName, SO.DateOrdered Order by TotalProfits desc	
6	What are the number of distinct products managed by each product manager?	select distinct E.EmployeeID, E.EmployeeFName, E.EmployeeLName, count(ProductName) as ProductsManaged from TEmployee E join TTerrainType TT on E.EmployeeID = TT.EmployeeID join TProduct P on TT.TerrainTypeID = P.TerrainTypeID group by E.EmployeeID, EmployeeFName, EmployeeLName	

	ProductName	TotalProfits	DateOrdered
1	Kestral R	5830	2017-10-28
2	Canis Double-Major	3728	2017-03-18
3	Boxer	3712	2017-07-19
4	Kestral R	3498	2017-01-15
5	Rotty	3311	2017-12-10
6	Raven	3164	2017-01-17
7	Kestral R	2915	2017-11-24
8	Osprey FX	2855	2017-08-17
9	Osprey FX	2855	2017-11-25
10	Rotty	2838	2017-08-16

	EmployeeID	EmployeeFName	EmployeeLName	ProductsManaged
1	43	Oleg	Dickson	5
2	65	Gray	Mcdaniel	3
3	82	Aubrey	Greer	3

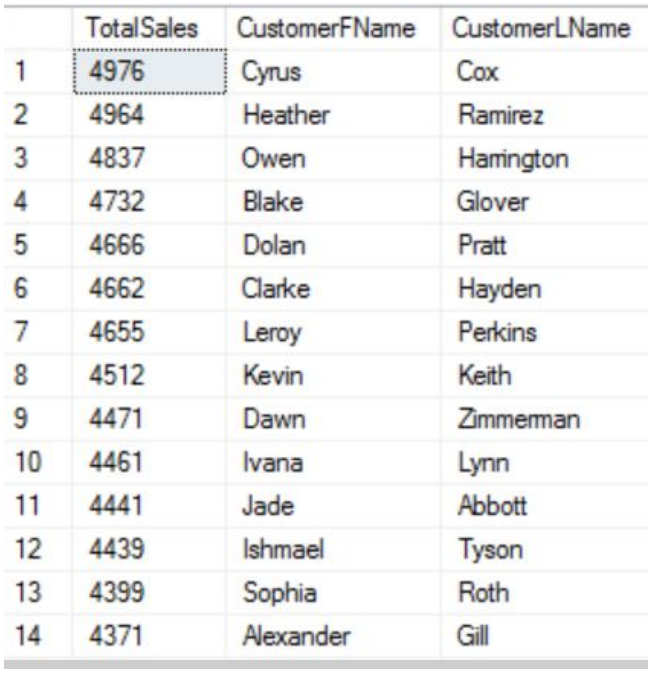
7	What are the total shipping costs for a given month by shipping type?	SELECT distinct DescriptionType, DeliveryOutDate, sum(ShippingCost) as TotalShipping FROM TDeliveryOut DO JOIN TDeliveryType DT ON DO.DeliveryTypeID = DT.DeliveryTypeID WHERE Month(DeliveryOutDate) = 04 GROUP BY DescriptionType, DeliveryOutDate	 DescriptionType DeliveryOutDate TotalShipping 1 In Person 2017-04-08 8 2 In Person 2017-04-17 8 3 In Person 2017-04-18 8 4 Normal 2017-04-20 5 5 In Person 2018-04-14 4 6 Normal 2018-04-15 6 7 Normal 2018-04-18 15 8 In Person 2018-04-28 5
8	How many sales order requests have been rejected by purchasing within a given year?	SELECT SaleOrReturn, count(SOLID) as AmountReturned FROM TSalesOrderLine SOL JOIN TSalesOrder SO ON SOL.SOID = SO.SOID WHERE SaleOrReturn = 'Return' and Year(DateOrdered) = 2017 GROUP BY SaleOrReturn	 SaleOrReturn AmountReturned 1 Return 75

9	What are the invoice lines for a given sales invoice number?	SELECT ProductName, Price, SOLTotal, Qty FROM TProduct P JOIN TSalesOrderLine SOL ON P.ProductID = SOL.ProductID	
10	What are all model-sizes and, for those that have been sold, how many sales of each has taken place?	select distinct L.LengthID, count(SOLID) as QtyOfProductSold, DescriptionLength From TLength L join TProduct P on L.LengthID = P.LengthID join TSalesOrderLine SOL on P.ProductID = SOL.ProductID Group by L.LengthID, DescriptionLength	

	ProductName	Price	SOLTotal	Qty
1	Thunderbird Bx	526	1833	4
2	Boxer	464	3114	3
3	Thunderbird Bx	526	4962	4
4	Kestral R	583	2160	4
5	Canis Double-Major	466	435	3
6	Thunderbird	515	1648	2
7	Osprey	414	4287	5
8	Canis Double-Major	466	3205	2
9	Raven	452	1506	4
10	Eagle	400	4497	3
11	Canis Double-Major	466	4618	4
12	Thunderbird	515	3623	3
13	Osprey	414	4328	4
14	Rotty	473	566	2

	LengthID	QtyOfProductSold	DescriptionLength
1	1	27	170
2	2	82	177
3	3	23	184
4	4	168	191

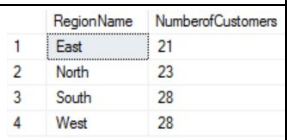
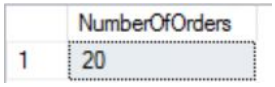
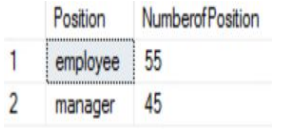
11	How many defects per product?	Select distinct P.ProductID, ProductName, count(SOLID) as Defects From TProduct P join TSalesOrderLine SOL on P.ProductID = SOL.ProductID Where SaleOrReturn = 'Return' Group by P.ProductID, ProductName	
12	A display of all ski products and their lengths.	SELECT ProductName, DescriptionLength FROM TProduct P Join TLength L ON P.LengthID = L.LengthID	
13	List of all customers that made a purchase within the last 3 months from the current date.	Select distinct CustomerFName, CustomerLName, DateOrdered From TSalesOrder SO join TCustomer C on SO.CustomerID = C.CustomerID where month(DateOrdered) between 10 and 12 and year(DateOrdered) = 2018	

		Group by CustomerFName, CustomerLName, DateOrdered	
14	List of customers whose average sales is less than the average of all sales.	SELECT sum(total) as TotalSales, CustomerFName, CustomerLName FROM TCustomer C JOIN TSalesOrder SO ON C.CustomerID = SO.SOID GROUP BY CustomerFName, CustomerLName ORDER BY TotalSales DESC	

	TotalSales	CustomerFName	CustomerLName
1	4976	Cyrus	Cox
2	4964	Heather	Ramirez
3	4837	Owen	Harrington
4	4732	Blake	Glover
5	4666	Dolan	Pratt
6	4662	Clarke	Hayden
7	4655	Leroy	Perkins
8	4512	Kevin	Keith
9	4471	Dawn	Zimmerman
10	4461	Ivana	Lynn
11	4441	Jade	Abbott
12	4439	Ishmael	Tyson
13	4399	Sophia	Roth
14	4371	Alexander	Gill

Three Additional Queries

It is important to have a variety of information to pull from to further a company. Sometimes what is asked is not enough to get the job done so it is good to always have more data if needed. Listed below are some additional queries that we felt would be beneficial towards helping company growth.

Query #	Question	Why is this important	SQL	Partial Output	Recap of Findings
1	How many customers are in each region.	It is important to know how many customers you have in an area so you can focus on large populations.	SELECT RegionName, count(CustomerID) as NumberofCustomers FROM TRegion R JOIN TCustomer C ON R.RegionID = C.RegionID GROUP BY RegionName		The largest amount of customers are in the South and the West.
2	How many sales orders placed in that month?	Helps Keep track of how many orders you are delivering per month	Select count(DateOrdered) as NumberOfOrders from TSalesOrder where MONTH(DateOrdered) = 3		The amount of Sales ordered made in the month of March was 20.
3	How many managers and employees are there?	It is good to know how many of each position you have in case you need to make employee adjustments.	SELECT Position, count(EmployeeID) as NumberofPosition FROM TEmployee GROUP BY Position		There are less managers than there are employees. This is good as you do not want the managers to outnumber the employees.

User Documentation

How to log into the database

1. First you will click on the search bar in the bottom left corner of your windows computer.
2. Then you will type in Remote Desktop Connection and click on it. The icon looks like this: 
3. After you have opened up Remote Desktop Connection you will enter your username and the computer name: rds.walton.uark.edu .
4. Then you will click on Show options
5. Click on Advanced/Settings and select "Use these RD Gateway settings".
6. Server Name: ent-sysgw.waltoncollege.uark.edu
7. Unclick "Bypass RD Gateway server for local addresses"
8. Go back to the general tab to press connect in the bottom right corner of the window.
9. You will then enter your username and password again.
10. No that you have accessed the database you will click on the Microsoft SQL Server application that looks like this: 

11. once you have opened up that you will type in the server name: essql1.walton.uark.edu ; make sure that the Server Type is database engine and the Authentication is Windows Authentication. Then click connect.
12. Click New Query on the left side of the applications toolbar.
13. Type in "Use ESa195331" into the white text box that appears after you complete step 12
14. You now have access to the database.

How to access data from a table

1. Type in "select * from [TABLE NAME YOU WANT TO ACCESS]
2. You can now see all the data for the table

How to update information in a table

1. Type in "Update [TABLE NAME]
2. Type set [attribute] = [value you want to change it to]
3. where [Primary Key] = [value of the primary key on the row you want to change]
   Example:

```
update TProduct
set WoodType = 'bamboo'
 where ProductID = 1
```

How to insert Data

1. Type in "Insert [TABLE NAME] (attributes name you want to insert)"
2. Type in "Values (values of attributes you wanna add in same order that you listed attributes by 'insert')"

Example:

```
Insert TProduct(ProductID, Length, Gender, TerrainType, Proficiency, ProductName, QtyOnHand, IsFlat, WoodType, TopSheetDesign, QtyCommitted, ReorderPoint, Cost, Price)
Values(22, 3, 1, 3, 1, 'Elmoskis', 6, 'Y', 'styrofoam', 'solid', 7, 1, 127, 586  )
select * from TProduct
```

What We Learned Throughout This Process

Through the process of creating an ERD, creating the logical design, learning and implementing the physical design, and understanding what is required of our team, Golden Retrievers has gained a wealth of knowledge about databases and teamwork. Utilizing various platforms such as LucidChart or Excel to organize our information have been imperative to our team's understanding of PSC's business.

Member Name:	**What you learned:**
Sophie Graham	Creating and organizing data is much easier said than done; databases are intended to organize data; through this team effort, I have learned manipulating data requires collaboration, curiosity, and patience. To successfully run a query, for example, the data has to be consistent with what the team establishes. As a team, we have to establish our goals and develop steps for how we will achieve them. Time management is part of the key to being successful within a group.
Trevor Doerr	Throughout this process I have learned technical things about ERD creation and running MySQL, as well as developed my soft skills of time management and communication with group members. Using LucidChart I created our teams model with the help and input of Sophie, Brandon and Sam and I learned a lot about how attributes and relations can be generalized in formats, but very specific when it comes to real customers. For example, adding a ship in and ship out table was specific to the customers' needs of recording employees that were checking and certifying both. Throughout the creation I had the opportunity to work on communication skills both explaining my reasoning for certain design aspects as well as listening to other ideas of teammates. Time management was definitely the most challenging part. I felt that our group met a sufficient amount of times and didn't have too many troubles scheduling those meetings, but I struggled in being prepared and staying focused some of those times which elongated the meeting for my whole group.
Sam Corley	The difficult part was learning all of the commands to implement everything. Once we had that down it wasn't too bad. We learned that creating a database is complicated and time consuming. There is a lot of components that go into making the database run correctly and a lot of troubleshooting. I learned that planning

	around everyone's schedule in a group project is more difficult than you would think. It's almost impossible to divide the workload evenly, but if everyone is doing their best to contribute and complete what they were assigned then it is a lot easier to work together.
Brandon Witte	While hard at the beginning adding data to a database gets easier the more and more you do it. Building a database has multiple steps to it and if you mess up one step along the way and don't realize it until later then you have to reset and go back and fix it. Running SQL statements is an effective way to get information from a large source of data. I am now better versed in using databases as well as using SQL in those databases. This project has taught me a lot about time management and about various other technical skills.

Appendix

Project Management

SPI Stands for Schedule Performance Index. If the value is less than one then it indicates that the project is potentially behind schedule whereas if the value is greater than one it indicates the project is most likely running ahead of schedule.

CPI stands for Cost Performance Index. It represents the amount of completed work for every unit spent. A CPI less than one means the project is over budget while a CPI greater than one means the project is running under budget.

Milestone 1:

SPI: 1.00
CPI: 4.00

Milestone 2:

SPI: 1.01
CPI: 6.40

Milestone 3:

SPI: 1.10
CPI: 6.30

Final:

SPI: 1.00
CPI: 4.00