Cereal Killers

Micayla Crow
Hayley Hunter
Brennan Parken
Margaret Medellin

Dr. Alexandra Durcikova

MIS 3353 – 002
Packwood Ski Company
Final Submission: Due 12/02/2018

Team Motto:
Never Miss Breakfast

Executive Summary

Our team offers efficient data management and design by implementing both best practices and meeting our client's needs. In this report, we seek to assist Packwood Ski Company by offering data solutions to help with tracking inventory, understanding major customers, improve product receiving, control purchasing processes, and employee success data. In addition, we worked to market penetration data, and offer various data for better decision making which allowed for major company improvements for Packwood Ski Company.

This report begins with interviewing the client, to ask about their requirements on the project. From this interview, we then assume that: all customers are invoiced the same way, customer information history is recorded, every customer has the same commission rate, a purchase order can be delivered over various deliveries, and that raw materials are recorded.

We then develop our Entity Relationship Diagram (ERD) for the project, as well as lists out the normalized relations from each table. Through the report, the group develops the ERD further to address data needs. For example, more entities were included to ensure more efficient and organized data. This helps differentiate and reduce data redundancy. Entities were also adjusted to make tracking attributes even easier, allowing more data to be accessible by the company to make informed business decisions.

Once we completed our Logical Design, which was an extremely important milestone for the project because we improved the integrity of our data, we moved on to the physical design. As a team, we were faced with many challenges we were forced to address. However, through the challenges we succeeded through the completion of all milestone stages.

As our project came to a close, we were able to evaluate our cost performance as a team with the project. We ended with a final SPI of 1.04 and a final CPI of .68. Our SPI is a high-performance index and means we planned for a decent number of budgeted hours to work. In addition, our CPI was a bit lower than expected. A CPI value of above 1 means that the project is doing well against the budget.

Table of Contents

Get to Know the Team: (Cereal Killers)

Team Members:

Name:	Major:	Graduate Year:	Internship Experience:	Background:
Hayley Hunter	*MIS*	*2019*	*None*	*Yukon, OK.*

Name:	Major:	Graduate Year:	Internship Experience:	Background:
Micayla Crow	*MIS*	*2019*	*Operations Intern – Private Equity at Blue River Partners, LLC*	*Tulsa, OK.*

Name:	Major:	Graduate Year:	Internship Experience:	Background:
Margaret Medellin	*MIS*	*2021*	*Analyst Intern - Medellin Applied Concepts Research*	*Dallas, TX.*

Name:	Major:	Graduate Year:	Internship Experience:	Background:
Brennan Parken	*MIS, Nonprofit Management*	*2019*	*Field Rep. - Oklahoma DeMolay*	*Yukon, OK.*

Conceptual Design

Within the conceptual design section, there is information about the Cereal Killer's first and last client meeting. In addition, all questions asked by the team as well as answers given for all of the questions by Ms. Sandberg will be found within this section. As the section wraps up, there will be extra information given to the team during the client meeting as Ms. Sandberg found it fit that the team was aware of information that was not explicitly given within the project's details.

The Client Meeting

Team 'Cereal Killers' had their first meeting with Ms. Sheryl Sandberg on October 15, 2018 at 1:00pm. The team members who were present to interview Ms. Sandberg were; Hayley Hunter, Margaret Medellin, and Micayla Crow. As the team sat down in Ms. Sandberg's office, we discussed any and all question we had in the creation of the project at hand (Packwood Ski Company). As our 15 minutes came to an end, our team closed the excellent meeting with nothing less than a professional thank you to Ms. Sandberg for allowing us the much-needed question and answer session to clear the air of any and all uncertainties.

Q&A During the Meeting & Information We Learned

In what order would you (Ms. Sandberg) like the database to present the entities and attributes (Example: alpha, no order, newest)? *"Alphabetical by last name."*

Is there any information that you (Ms. Sandberg) would want hidden from the user or would you like us to recall all information within the selected attribute (example: give old names/numbers)? *"Be able to recall all old information. This information could be useful when keeping track of wholesalers in case they move companies."*

Do the companies the client's in negotiation with have different currency other than Canada or US? *"No, the currency will be the same."*

Are there specific payment terms based on size and relationships that you (Ms. Sandberg) would like to integrate into the entity relationship diagram? *"We would like to incorporate balance forward and invoice open for orders $50,000 or below. However, the salespeople have to decide additional payment terms based on size and relationship to give to a given customer."*

Do the managers have multiple employees? If so, can an employee work for multiple managers? *"One manager has multiple employees and therefore multiple employees have one manager. In addition, one manager can manage multiple categories."*

Would you (Ms. Sandberg) like to create a reorder point for your raw materials or would this be hard with the changing of vendors you have? *"No, we do not need a reorder point, but we will need to know quality of raw materials from our one supplier. We batch order seasonally online and only keep a sample of each model over years so there is no inventory kept or recorded."*

When categorizing the products of the entity relationship diagram, you (Ms. Sandberg) mentioned that "item" refers to a particular model in a particular length when tracking the items

ordered. Earlier in the reading you asked to categorize the products by gender, type, proficiency, etc. Is there a way you would like to display the product categories for viewing product order? *"When viewing the product order, we want to know all about the order details. So, it is necessary to have each order associated with type, proficiency, gender, length, model, etc."*

Would you (Ms. Sandberg) like for us to create the entity relationship diagram to show when upper management decides to waive or alter the shipping cost? *"Yes, Keep track. Also keep track of approval. Salespeople equal work on commision so this is important. We should know which account to charge it to as well. Discount is also important in this, need to know who approved it and exceptions accounts to charge to."*

During the manufacturing and sales processes, can an employee manage multiple product categories or just one? *"Yes, employees can manage multiple product categories. (Multiple can mean just one or two or it can be a lot)."*

The packet explains the desire for the team to provide three more reports that could be beneficial. Would a report from the data that would break down a loss or gain in profit each month based on region be desired? Data from sales and costs reports would be used. *"Yes, we need to get as many metrics as possible & any info is helpful."*

CLIENT GIVES EXTRA INFO.

"Track which customers purchased on a regular basis. Accounts receivable in good standing. Terms agreed should be based on Net 30/45/60."

"Need to know current contact person history if they change companies."

"Need to incorporate the sample data."

"Shipping needs to be tracked if it's picked up, shipped, order received, carrier, bill of shipping numbers."

"Track customer orders and payments."

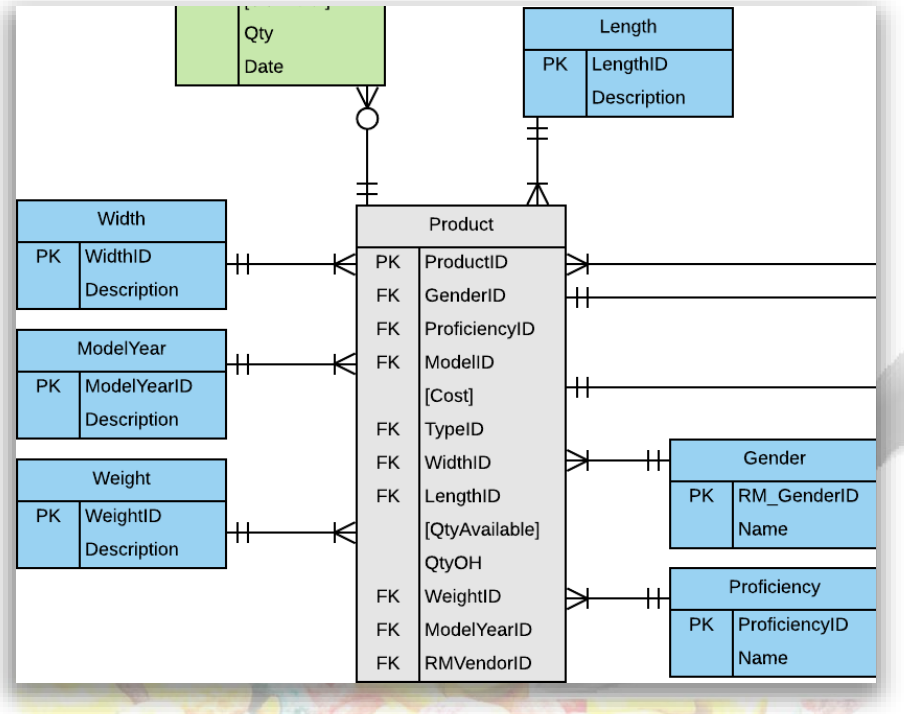
"The system needs to know when and how much discount will be applied."

"Products are ordered by season."

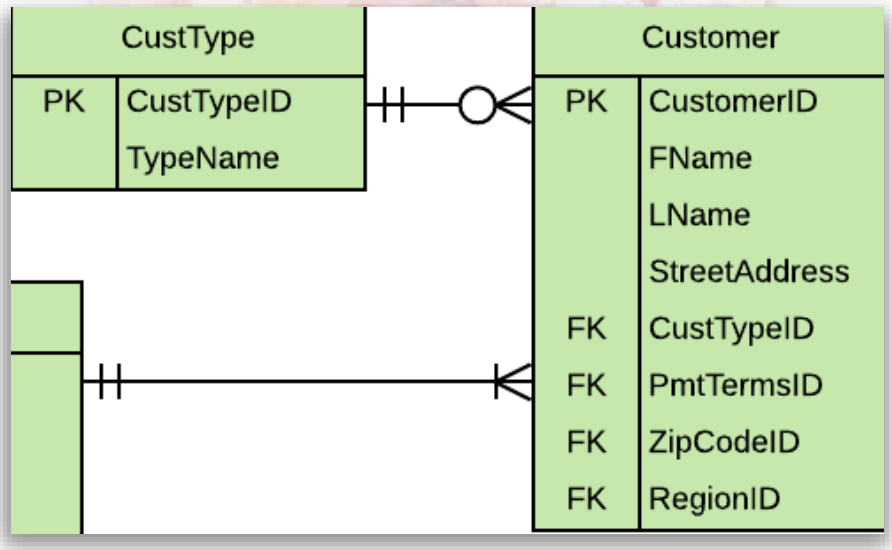
Significant Assumptions

Though we received excellent answers from Ms. Sandberg for all the questions we had, as we began the project new questions arose. As we addressed the newfound questions, we made assumptions in regard to specific relationships as well as the possible outcomes. Within the significant assumptions section, we have presented the assumptions made along with images of where they are reflected within the entity relationship diagram.

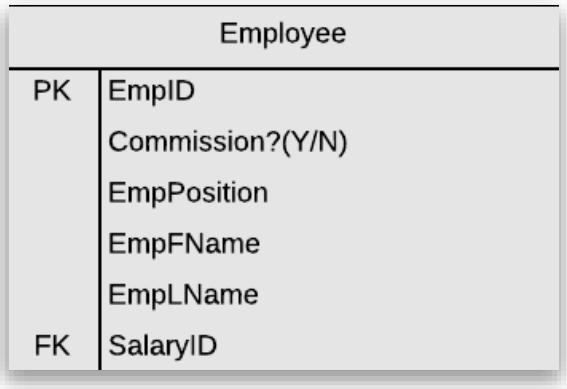
Assumption 1 - Where can we keep product measurements?
Answer: *We can keep product measurements in entities such as: Length, Width, and Weight.*



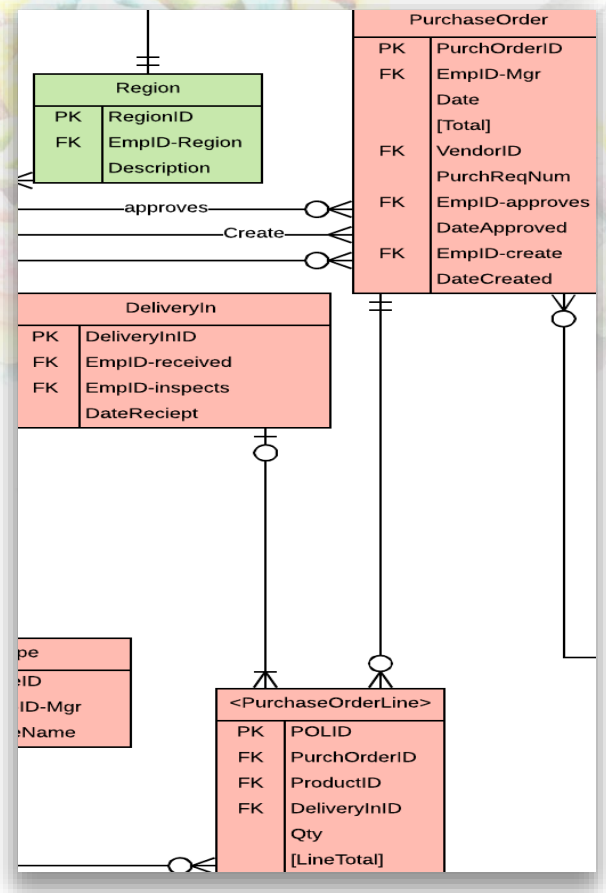
Assumption 2 - Where can we keep customer type?
Answer: *We can keep customer type within the CustType entity.*



Assumption 3 - Are all commissioned employees paid the same commission rate?
Answer: *Yes, as shown by the commission attribute within the Employee entity.*

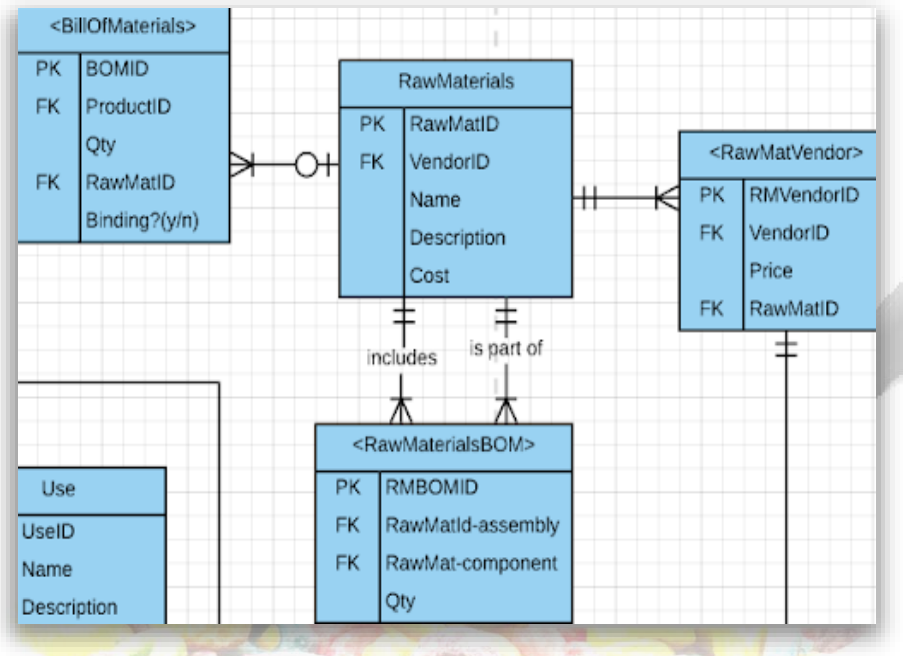


Assumption 4 - Can a purchase order be delivered over multiple deliveries?
Answer: *Yes, as shown in the relationship between DeliveryIn and <PurchaseOrderLine>, this allows purchase orders to be delivered over multiple deliveries (otherwise, the relationship would be between DeliveryIn and PurchaseOrder).*



Assumption 5 - Does Packwood Ski Company keep track of their Raw Materials needed for the Raw Materials used in the product?

Answer: *Yes, in the Raw Materials Bill of Materials. This accounts for the unfinished materials that need assembly to be a Raw Material. The two one and only one to zero to many cardinalities between <RawMaterialsBOM> and RawMaterials allow for this storage.*



Entity Relationship Diagram

The entity relationship diagram section includes information as to what an entity relationship diagram is and why it is necessary. In addition, this section will include the business cycles used within the diagram Revenue Cycle, Expenditure Cycle, and Production Cycle. We have also added an image of the first diagram made with the business cycles colored Revenue Cycle (green), Expenditure Cycle (red), and Production Cycle (blue). Lastly, this section will have descriptions of the changes we have made to our entity relationship diagram to better suit the requests of the client (Packwood Ski Company).

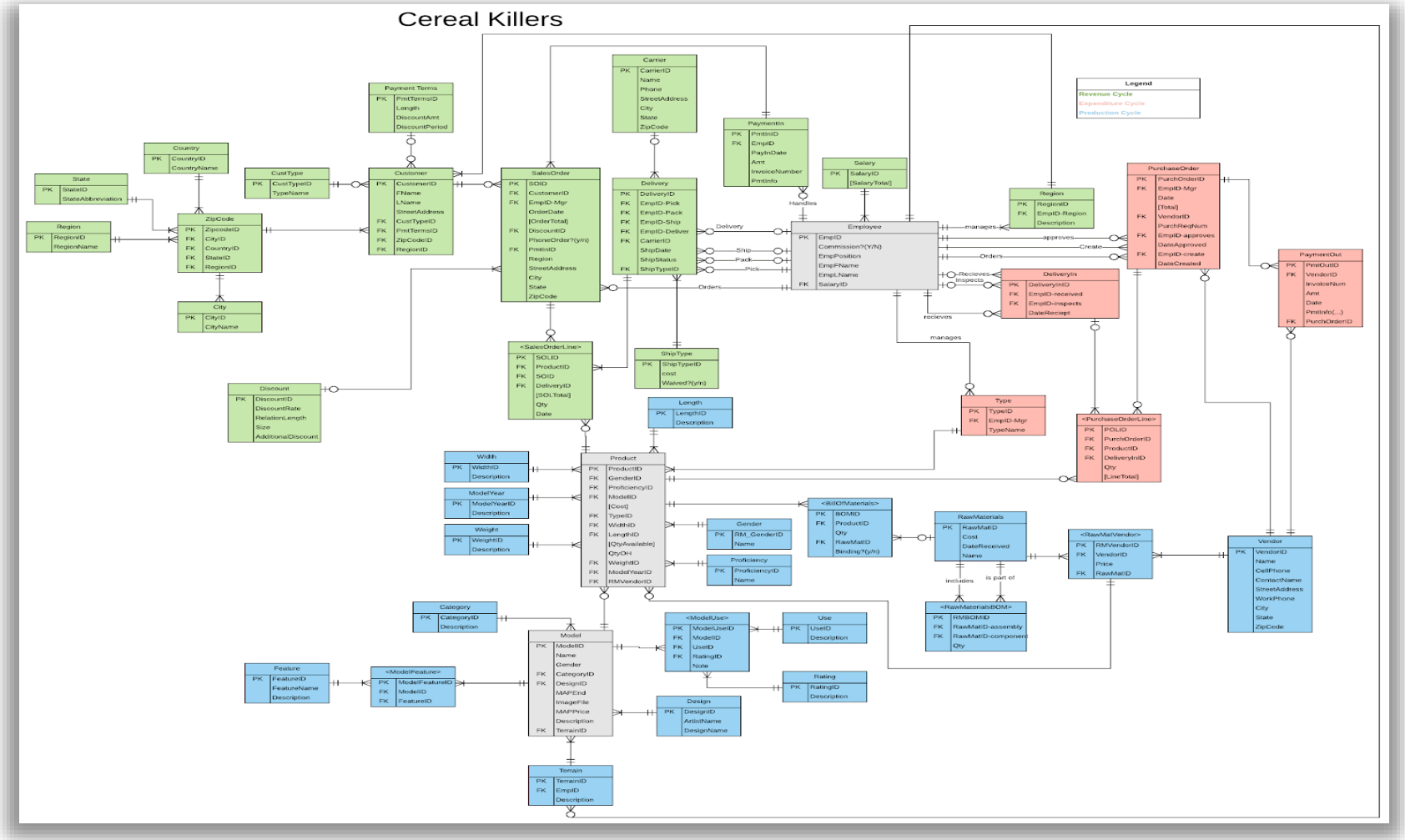
What is an Entity Relationship Diagram? Why is it necessary?

Think of a house. You know all of the things you want it to consist of such as a swimming pool, five bedrooms, kitchen, etc. The entity relationship diagram would serve as a blueprint to building your house in this situation. It is a representation of the information specified to build your desired house. It is the framework of tables whom have visual representations of the relationships between entities (object, person, or concept) in a database that contain special information about the house and its processes. A database is basically a place where all of this information is stored. Each entity consists of attributes which are data about the entity. In conclusion, entity relationship diagrams are extremely important and useful in the discovery of how the database receives, sets, and where it pulls information from.

Business Cycles Used

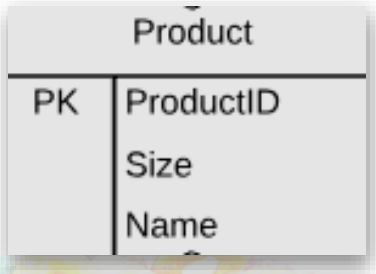
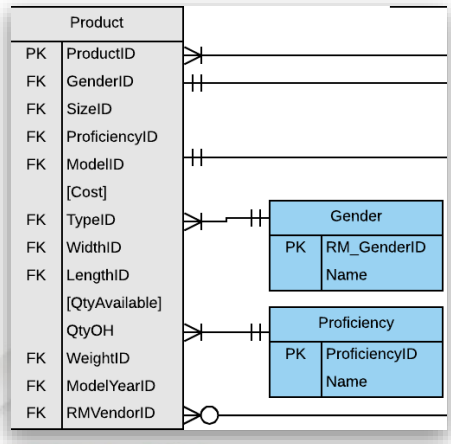
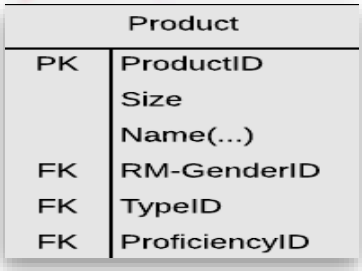
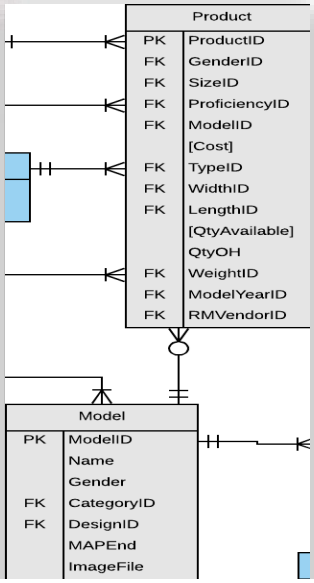
The Accounting Cycle entity relationship diagrams we used were the Revenue Cycle, Expenditure Cycle, and Production Cycle.

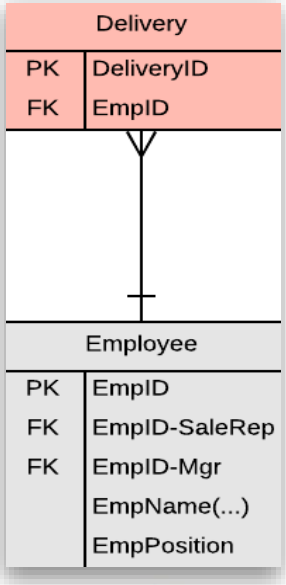
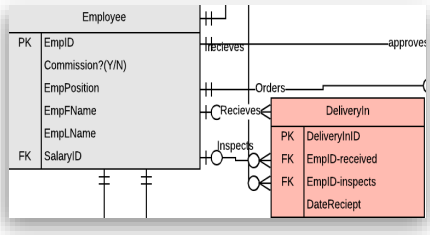
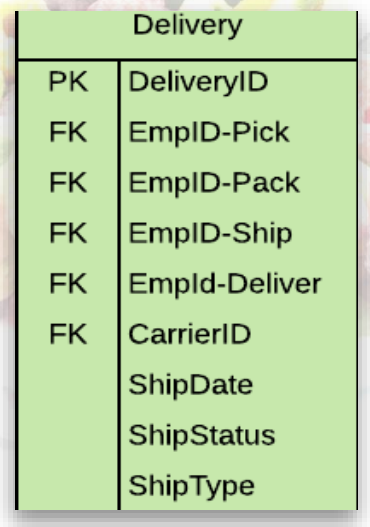
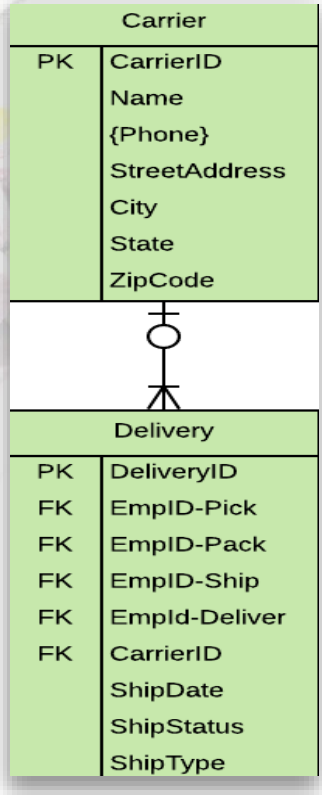
Entity Relationship Diagram Created

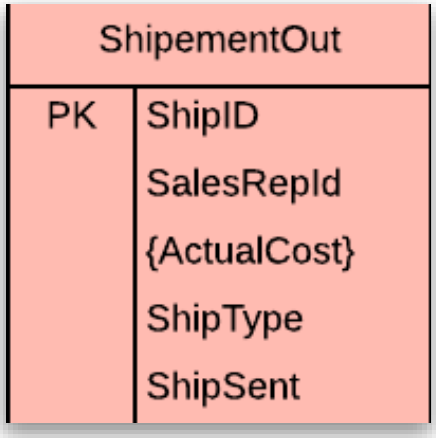
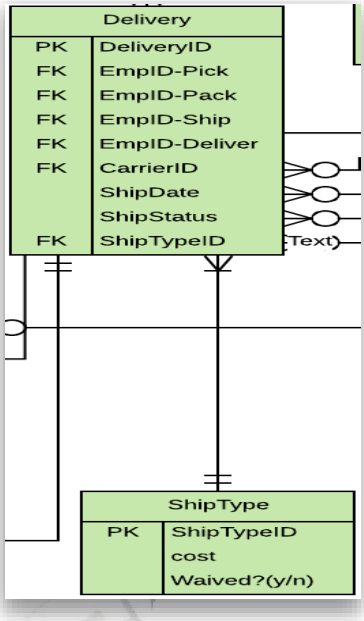
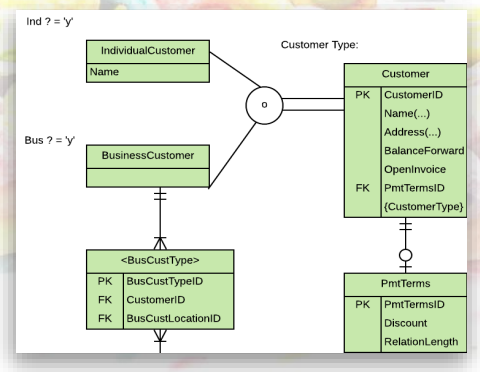
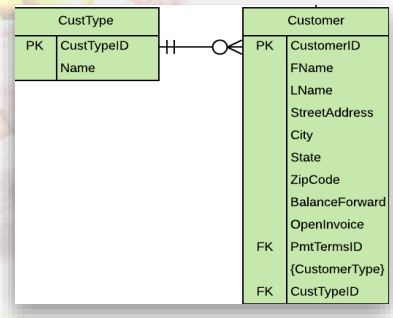


Changes made to generic Entity Relationship Diagrams

Below are the changes we have made to our respective entity relationship diagram. In the display, you will find pictures of the original entity relationship diagram then to the right you will see pictures of the updated entity relationship diagram, as we have made the transcribed changes.

Change #	Original ERD	Updated ERD
The first major change was adding the "Proficiency" and "Gender" entities. These are not typically a part of the generic entity relationship diagrams. These were necessary because these are extra categories that Packwood specified, they wanted us to track. In a sense, they are extra "product categories" that must have their own entities in order for all of the desired data to be included. It also makes the data more efficient and organized.		
We added the Model entity that has a relationship with Product entity because Packwood explains that each design varies by model. We did this so Packwood could keep better track of attributes such as gender, year, category, as well as price of the products.		

Employee and DeliveryIn entities have two relationships: inspecting and receiving. This is shown in our cardinalities in the entity relationship diagram. Typically, there is only one relationship between these two entities.		
We added a Carrier entity because there are multiple types of delivery methods that can be used. These types include overnight, two-day, freight, or hand delivered and each have a different cost. The relationships are shown between entities Employee, Delivery, and Carrier.		

We have an extra ShipType entity to better clarify the way the product is being shipped.		
CustType is not typically included in a generic entity relationship diagram. The reason we have it is because Packwood says they only sell to wholesalers and retailers, never directly to customers. PSC needs to know CustType to know when and to who to discount.		

Logical Design

Logical Design is good for new construction within projects because it is one step closer to implementation. In addition, logical design is good for fixing bad databases (databases with no integrity) as well as separating content from the structure to enable building of an efficient, accurate database (one without duplicates). The logical design has terminology such as Relations, Columns, and Rows. Within the logical design section is information about normalization. Following the logical design section is the normalized relation section which shows the relations between the entities.

Normalization

Normalization is a process that developers implement to help ensure that a database is trustworthy or more specifically, has data integrity and the reports generated reflect reality. Through Normalization, the reports generated are also efficient, meaning, they use minimal time to receive information from the database. Essentially, if anyone cares about Six Sigma experiences when interacting with databases, then normalization is highly prioritized.

Normalization requires two things: Ensure atomicity of columns (meaning that the field values cannot be reasonably broken down any further), and Eliminate data duplication issues (meaning that field values do not have duplications that would render the ability to properly read data. This occurs when values for a given field are the same across multiple records within a given table).

There are multiple stages of normalization or Normal Forms. For the sake of time and understanding of each of the normal forms, our class will only cover normal forms zero to three. Normal form zero is the first normal form. A relation is written in 0NF if it is something that exists and is not atomic whatsoever. A relation is written in 1NF if it is atomic (meaning that there is no multi-value or composite attributes) and if it has partial functional dependencies. In addition, 2NF is when there are no partial functional dependencies and there are transitive dependencies. Last but not least, 3NF is when there are no transitive dependencies. We want out database to be in 3NF because it allows for data integrity. With a database that is in 3NF we can be confident that there are no duplications that could risk the reliability of the information being pulled from the database.

Normalized Relations

Below are the normalized relations in third normal form, as they are represented within the entity relationship diagram.

Revenue Cycle:

TCustType (CustTypeID, CustTypeName)

TCustomer (CustomerID, CustFName, CustLName, CustStreetAddress, PmtTermsID,

CustTypeID, ZipCodeID, RegionID)

Foreign Key PmtTermsID refers to TPaymentTerms
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key CustTypeID refers to TCustType
Not Null
On Delete Restrict
On Update Cascade

Foreign Key ZipCodeID refers to TZipCode
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RegionID refers to TRegion
Null Allowed
On Delete Set Null
On Update Cascade

TPaymentTerms (PmtTermsID, PmtLength, PmtDiscountAmt, PmtDiscountPeriod)

TSalesOrder (SOID, CustomerID, EmpID-Mgr, DiscountID, PmtInID, SOOrderDate, SOOrderTotal, SOPhoneOrder, SOStreetAddress, SOCity, SOState, SOZipCode, SORegion)

Foreign Key CustomerID refers to TCustomer
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID-Mgr refers to TCustomer
Not Null
On Delete Restrict
On Update Cascade

Foreign Key DiscountID refers to TDiscount
Not Null
On Delete Restrict
On Update Cascade

Foreign Key PmtInID refers to TPaymentIn
Not Null
On Delete Restrict
On Update Cascade

TSalesOrderLine (SOLID, ProductID, SOID, DeliveryID, SOLSOLTotal, SOLQty, SOLDate)

Foreign Key ProductID refers to TProduct
Not Null

On Delete Restrict
On Update Cascade

Foreign Key SOID refers to TSaleOrder
Not Null
On Delete Restrict
On Update Cascade

Foreign Key DeliveryID refers to TDelivery
Not Null
On Delete Restrict
On Update Cascade

TDelivery (DeliveryID, EmpID-Pick, EmpID-Pack, EmpID-Ship, EmpID-Deliver, CarrierID, ShipTypeID, DelShipDate, DelShipStatus)

Foreign Key EmpID-Pick refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID-Pack refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID-Ship refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID-Deliver refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key CarrierID refers to TCarrier
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key ShipTypeID refers to TShipType
Null Allowed
On Delete Set Null
On Update Cascade

TCarrier (CarrierID, CarName, CarPhone, CarStreetAddress, CarCity, CarState, CarZipCode)

TSalary (SalaryID, SalSalaryTotal)

TEmployee (EmpID, EmpCommission, EmpPosition, EmpFName, EmpLName, SalaryID)

Foreign Key SalaryID refers to TSalary
Null Allowed
On Delete Set Null
On Update Cascade

TProduct (ProductID, GenderID, ProficiencyID, ModelID, WidthID, LengthID, TypeID, WeightID, ModelYearID, RMVendorID, PCost, PQtyAvailable, PQtyOH)

Foreign Key GenderID refers to TGender
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key SizeID refers to TSize
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key ProficiencyID refers to T Proficiency
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key ModelID refers to TModel
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key WidthID refers to TWidth
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key LengthID refers to TLength
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key TypeID refers to TType
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key WeightID refers to TWeight
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key ModelYearID refers to TModelYear
Null Allowed
On Delete Set Null

On Update Cascade

Foreign Key RMVendorID refers to TRawMatVendor
Null Allowed
On Delete Set Null
On Update Cascade

TShipType (ShipTypeID, ShipCost, ShipWaived)

TPaymentIn (PmtInID, EmpID, PayPayInDate, PayAmt, PayInvoiceNumber, PayPmtInfo)

Foreign Key EmpID refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

TRegion (RegionID, EmpID-Region, RDescription)

Foreign Key EmpID-Region refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

TDiscount (DiscountID, DDiscountRate, DRelationLength, DSize, DAdditionalDiscount)

TCountry (CountryID, CCountryName)

TState (StateID, SStateAbbreviation)

TRegion (RegionID, RRegionName)

TCity (CityID, CityCityName)

TZipCode (ZipCodeID, CityID, CountryID, StateID, RegionID)

Foreign Key CityID refers to TCity
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key CountryID refers to TCountry
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key StateID refers to TState
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RegionID refers to TRegion
Null Allowed

On Delete Set Null
On Update Cascade

Production Cycle:

TGender (RM-genderID, GName)

TProficiency (ProficiencyID, PName)

TBillOfMaterials (BOMID, ProductID, BOMQty, RawMatlID, BOMBinding)

Foreign Key ProductID refers to TProduct
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RawMatlID refers to TRawMaterials
Null Allowed
On Delete Set Null
On Update Cascade

TRawMaterials (RawMatID, RMName, RMDateRecieved, RMCost)

TRawMaterialsBOM (RMBOMID, RawMatID-assembly, RawMatID-component, RMBOMQty)

Foreign Key RawMatid-assembly refers to TRawMaterials
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RawMat-component refers to TRawMaterials
Null Allowed
On Delete Set Null
On Update Cascade

TVendor (VendorID, VName, VCellPhone, VWorkPhone, VContactName, VStreetAddress, VCity, VState, VZipCode)

TModel (ModelID, MName, MGender, CategoryID, TerrainID, MMAPEnd, MMAPPrice, MDescription, MImageFile, DesignID)

Foreign Key CategoryID refers to TCategory
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key TerrainID refers to TTerrain
Null Allowed

On Delete Set Null
On Update Cascade

Foreign Key DesignID refers to TDesign
Null Allowed
On Delete Set Null
On Update Cascade

TModelUse (ModelUseID, ModelID, UseID, RatingID, MUNote)

Foreign Key ModelID refers to TModel
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key UseID refers to TUse
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RatingID refers to TRating
Null Allowed
On Delete Set Null
On Update Cascade

TUse (UseID, UDescription)

TRawMatVendor (RMVendorID, VendorID, RawMatID, Price)

Foreign Key VendorID refers to TVendor
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key RawMatID refers to TRawMaterials
Null Allowed
On Delete Set Null
On Update Cascade

TRating (RatingID, Description)

TDesign (DesignID, DesignArtistName, DesignDesignName)

TTerrain (TerrainID, EmpID, TerrainDescription)

Foreign Key EmpID refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

TModelFeature (ModelFeatureID, ModelID, FeatureID)

Foreign Key ModelID refers to TModel
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key FeatureID refers to TFeature
Null Allowed
On Delete Set Null
On Update Cascade

TFeature (FeatureID, FFeatureName, FDescription)

TCategory (CategoryID, CategoryDescrption)

TWeight (WeightID, WDescription)

TModelYear (ModelYearID, ModelYearDescription)

TWidth (WidthID, WidthDescription)

TLength (LengthID, LengthDescription)

Expenditure Cycle:

TPurchaseOrder (PurchOrderID, EmpID-Mgr, PurchDate, PurchTotal, VendorID, PurchPurchReqNum, EmpID-approves, PurchDateApproved, EmpID-create, PurchDateCreated)

Foreign Key EmpID-Mgr refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key VendorID refers to TVendor
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID-approves refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

Foreign Key EmpID- create refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

TDeliveryIn (DeliveryInID, EmpID-received, EmpID-inspects, DelInDateReciept)

Foreign Key EmpID-received refers to TEmployee
Null Allowed

On Delete Set Null
On Update Cascade

Foreign Key EmpID- inspects refers to TEmployee
Null Allowed
On Delete Set Null
On Update Cascade

TPurchaseOrderLine (POLID, PurchOrderID, ProductID, POLQty, POLLineTotal, DeliveryInID)

Foreign Key PurchOrderID refers to TPurchaseOrder
Null Allowed
On Delete Set Null

Foreign Key ProductID refers to TProduct
Null Allowed
On Delete Set Null

Foreign Key DeliveryInID refers to TDeliveryIn
Null Allowed
On Delete Set Null

TPaymentOut (PmtOutID, VendorID, POInvoiceNum, POAmt, PODate, POPmtInfo, PurchOrderID)

Foreign Key VendorID refers to TVendor
Null Allowed
On Delete Set Null

Foreign Key PurchOrderID refers to TPurchaseOrder
Null Allowed
On Delete Set Null

TType (TypeID, EmpID-Mgr, TypeTypeName)

Foreign Key EmpID-Mgr refers to TEmployee
Null Allowed
On Delete Set Null

Differences between Entity Relationship Diagram and Normalized Relations

The entity relationship diagram is the first blueprint that a client sees. It is like a layout of a house. However, the entity relationship diagram needs to be logically designed and then normalized in order to be implemented into a database correctly. There are 3 Normal Forms that organizes fields and tables of a Relational Database Management System (RDBMS) to ensure well-structured relations and we peak at the third. 3NF has no transitive dependencies (non-key attribute predicts other non-key attributes) and no partial functional dependencies (non-key attribute(s) predicted by fewer than all key attributes). The benefits of normalized relations

include ensuring atomic columns and eliminating data redundancy. By normalizing an entity relationship diagram relation animalities such as deletion, insertion, and modification can be avoided. For example, when transforming our entity relationship diagram into normalized relations, we had to change the multi-valued attribute "Name(…)" is, into "FirstName, LastName" to ensure no data redundancy.

The entity relationship diagram we created and the Normalized Relations where very similar, however the Normalized relations we picked helped us decide better uses of our cardinality choices, additions of entities, and attribute placements. The most influential impact the Normalized Relations had in my opinion, was how it helped show the relationships of foreign keys to primary keys. This helped us realize if we needed to change any cardinality issues and made things easier to spot that could potentially cause issues.

Referential Integrity

"Referential" means an association among entities. The integrity constraint means that if there is a relationship, the foreign key must match a valid primary key. The only exception is if it is null. As far as the entity goes, a primary key must always exist and cannot be null. All values in a column must be from the same "domain". This basically means that they are drawn from a set of allowable values/value types specified for that column. We show this through normalization in our logical designs that we are attaching here.

Physical Design and Implementation

Physcial design allows for efficient or fast implementations. This section will present a table of our physical design along with information to better understand data dictionaries, and denormalization. In addition, this section will cover the challenges our team faced and addressed during implementation, as well as the strengths and weaknesses we faced and addressed.

Data Dictionary

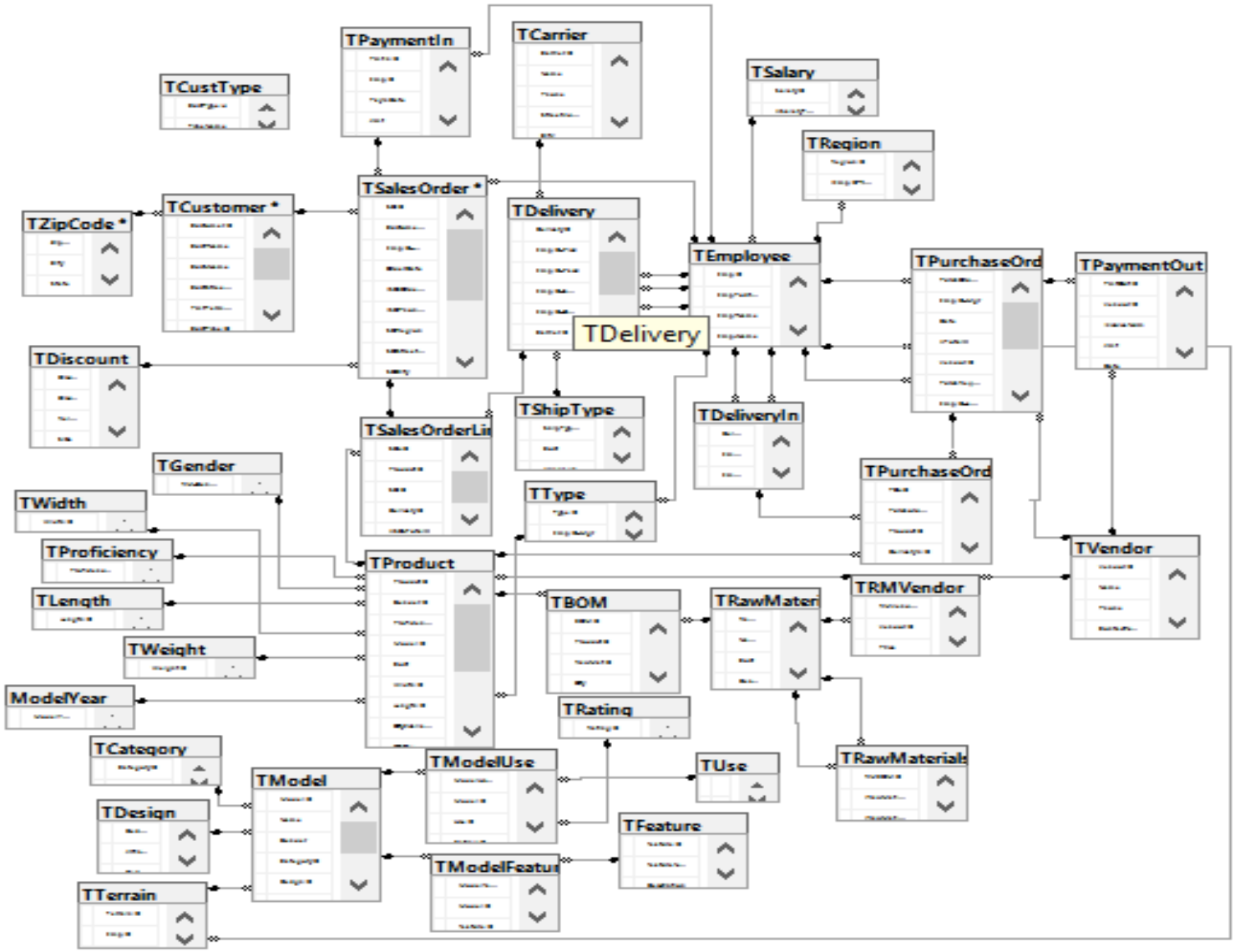
A data dictionary contains records about other objects and entities in the database, such as data relationships and data ownership. The data dictionary provides a breakdown of data types and references, making it an easier way to fully grasp what the database is trying to communicate. It breaks down each single attribute meaning as well as the allowable values. Our dictionary model is located within the appendex.

Denormalization

Denormalization is often used to enhance the efficiency through physical design. Denormalization refers to the reducing of normalization (i.e., diminishing the normal form) of a table or tables within a database usually by allowing data duplication to occur (denormalizing could also refer to reducing atomicity in a field or fields, but this is rarely done, so we're going to ignore this form of denormalization going forward). Traditionally, though, denormalization

has been seen as a way to allow a database to process queries more quickly. Since every additional table that must be referenced in a query will make the query take a little more time to complete, reducing the number of tables needed has been seen as a potential positive. Doing so has always, however, introduced problems with data integrity by allowing data duplication.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

One of the biggest challenges we faced during the creation of the database was our misunderstanding of how to save changes witin the SQL server. There were times the team had to go in and recreate all the diagrams because there were issues with saving the work. We addressed this issue by assigning our foreign keys correctly. The second biggest challenge we faced, as a team, was finding each others strengths and weaknesses then splitting the work so it could be done as timely and efficiently as possible.

Strengths and Weaknesses Encountered During Implementation

A strength of our team included adding the data tables to the right database name and logging hours in project MIS more frequently. Our main weakness was differentiating each group members part because there were a lot of intertwining parts. Often, there were times team members had to go in and check eachothers work for inaccuracies that came about due to the different styles we all had in completing work. There was not a single way to complete our tasks correctly, therefore, we had to learn eachothers way of thinking so that we could build off of our members work.

Specific SQL Statements Requested

Below are the queries we have developed in SQL from the client's requests and a few additional queries we thought would be useful as well.

Query #	Question	SQL	Partial Output
1	Total sales (in dollars) by region in a given month (note that we would like to be able to input the month to be calculated)	**SELECT** R.RegionID, S.SOID, S.OrderDate, SUM(S.[SO(OrderTotal)]) AS [Total Sales] **FROM** TSalesOrder S, TCustomer C, TRegion R **WHERE** R.RegionID = C.RegionID AND C.CustomerID = S.CustomerID AND OrderDate Between '10/01/2018' AND '10/31/2018' **GROUP BY** R.RegionID, S.SOID, S.OrderDate;	Results Messages RegionID SOID OrderDate Total Sales 1 5 40 2018-10-20 548.73 2 5 50 2018-10-20 599.42 3 2 62 2018-10-25 178.69 4 1 76 2018-10-01 506.61 5 3 93 2018-10-28 278.36

2	Total sales (in dollars) by customer in a given year as well as total expenditure by vendor in a given year . (This should be two separate reports.)	**SELECT** C.CustomerID, C.CustFName, S.OrderDate, SUM(S.[SO(OrderTotal)]) AS [TotalSalesPerCust] **FROM** TSalesOrder S, TCustomer C **WHERE** C.CustomerID = S.CustomerID AND S.OrderDate Between '01/01/2018' AND '12/31/2018' **GROUP BY** C.CustomerID, C.CustFName, S.OrderDate;	Results / Messages CustomerID, CustFName, OrderDate, TotalSalesPerCust 1 31 Melanie 2018-01-05 528.42 2 97 Irene 2018-01-11 314.88 3 37 Maggie 2018-01-14 12.78 4 28 Xyla 2018-01-19 17.88 5 56 Dennis 2018-03-02 743.03 6 26 James 2018-03-10 488.72 7 87 Thor 2018-03-11 48.86 8 18 Myles 2018-03-24 902.92
3	Total sales (in dollars) by customer in a given year as well as total expenditure by vendor in a given year . (This should be two separate reports.)	**SELECT** V.VendorID, V.Name, SUM(M.Price) As [TotalExpenPerVen] **FROM** TVendor V, TRawMaterial2 R,TRMVendor M WHERE R.RawMatID = M.RawMatID AND V.VendorID = M.VendorID AND R.DateReceived Between '01/01/2018' AND '12/31/2018' **GROUP BY** V.VendorID, V.Name;	Results / Messages VendorID, Name, TotalExpenPerVen 1 1 Aenean Company 4935.00 2 2 Nec LLP 97.00 3 4 Mauris Blandit Mattis Limited 1793.00 4 5 Sed Foundation 176.00 5 10 Sed Pharetra Incorporated 2948.00 6 12 Malesuada Limited 4456.00 7 15 Sed Hendrerit Institute 2774.00 8 17 Tempor Bibendum Incorporated 4763.00
4	The ten highest selling (a) models, (b) terrain ski types, and (c) model -sizes.	**Select** top 10 P.ProductID, Sum(QtyAvailable) As [ModelOrdered], Name **from** TProduct p, TSalesOrderLine S, TModel M **Where** P.ProductID = S.ProductID AND M.ModelID = P.ModelID **Group by** P.ProductID, Name **order by** ModelOrdered DESC;	Results / Messages ProductID, ModelOrdered, Name 1 22 996 PackBadger 2 65 984 Boxer 3 57 978 Eagle 4 86 964 Kestral R 5 23 959 Kyon 6 40 947 Kestral R 7 33 947 Osprey 8 41 931 Kestral M 9 72 927 Raven 10 36 920 Rotty
5	The ten highest selling (a) models, (b) terrain ski types, and (c) model -sizes.	**SELECT** Top 10 T.TerrainID, Sum(Qty) AS [Ordered], M.Name, M.ModelID **FROM** TTerrain T, TModel M, TProduct P, TSalesOrderLine L **WHERE** T.TerrainID = M.TerrainID And M.ModelID = P.ModelID And P.ProductID = L.ProductID	Results / Messages TerrainID, Ordered, Name, ModelID 1 8 74 Osprey 5 2 2 59 PackBadger 4 3 9 50 Rotty 6 4 2 48 Thunderbird 7 5 8 47 Raven 2 6 9 43 Kyon 9 7 9 41 Boxer 3 8 8 40 Thunderbird Bx 8 9 9 38 Kestral R 12 10 2 38 Eagle 1

		GROUP BY M.Name, M.ModelID, T.TerrainID **ORDER BY** Ordered DESC;	
6	The ten highest selling (a) models, (b) terrain ski types, and (c) model -sizes.	**SELECT** Top 10 Sum(Qty) AS [Ordered], M.Name, M.ModelID,T.Description as [Length] **FROM** TProduct P, TSalesOrderLine L, TModel M, TLength T **WHERE** P.ModelID = M.ModelID AND P.ProductID = L.ProductID AND P.LengthID = T.LengthID **GROUP BY** T.Description, Name, M.ModelID **ORDER BY** Ordered DESC;	Results / Messages Ordered, Name, ModelID, Length 1 32 Osprey 5 170 2 30 PackBadger 4 184 3 30 Osprey 5 191 4 25 Kestral R 12 191 5 25 Thunderbird 7 177 6 24 Canis Double-Major 11 177 7 21 Rotty 6 191 8 21 Kyon 9 191 9 20 Thunderbird Bx 8 184 10 19 Boxer 3 184
7	Purchase order requests (product manager name, purchase req number, item, quantity) that have been rejected by purchasing within a given year.	**SELECT** P.EmpID_Mgr, EmpFName + ' ' + EmpLName **AS** [Requested], P.PurchOrderID, L.ProductID, L.Qty **FROM** TEmployee E, TPurchaseOrder P, TPurchaseOrderLine L **WHERE** E.EmpID = P.EmpID_Mgr **AND** L.PurchorderID = P.PurchOrderID **AND YEAR**(Date)= 2018;	Results / Messages EmpID_Mgr, Requested, PurchOrderID, ProductID, Qty 1 1 Suki Clark 1 1 8638 2 2 Amethyst Mills 2 2 2116 3 5 Madaline Dudley 5 5 6272 4 7 Allegra Walker 7 7 9738 5 10 Kimberley Ruiz 10 10 2457 6 15 Vaughan Mcclure 15 15 841
8	Invoice lines for a given sales invoice number (i.e., the quantity, product number, product name, price, and line total for each product sold as part of a given order)	**SELECT** P.ProductID, M.Name, L.Qty, M.MAPPrice, M.MAPPrice*Qty **AS** [LineTotal] **FROM** TProduct P, TModel M, TSalesOrderLine L **WHERE** P.ModelID = M.ModelID AND P.ProductID = L.ProductID;	Results / Messages ProductID, Name, Qty, MAPPrice, LineTotal 1 1 PackBadger 9 249.99 2249.91 2 2 Osprey 2 609.99 1219.98 3 3 PackBadger 4 249.99 999.96 4 4 Osprey 9 609.99 5489.91 5 5 Kyon 4 649.99 2599.96
9	All model -sizes (regardless of whether a model	**SELECT DISTINCT** M.Name **AS** [ModelName], p.ProductID, W.Description **AS** [Size], O.Qty **AS** [QTYSOLD]	

	-size has been sold) and, for those that have been sold, how many sales of each has taken place.	**FROM** TModel M, TProduct P, TPurchaseOrderLine O, TLength L, TWidth W **WHERE** M.ModelID=P.ModelID **AND** P.ProductID=O.ProductID	

	ModelName	ProductID	Size	QTYSOLD
1	Thunderbird	10	108x81x103	2457
2	PackBadger	13	129x102x124	2134
3	Kyon	18	123x98x118	6102
4	Thunderbird Bx	28	129x112x120	2521
5	Canis Double-Major	39	142x111x137	6756

10	Total shipping costs for a given month by shipping type (freight, two -day, overnight).	**SELECT SUM** (S.Cost) **AS** [TotalShippingPrice], S.Description **FROM** TShipType S, TDelivery D **WHERE** S.ShipTypeID=D.ShipTypeID **AND** **MONTH**(D.ShipDate) = 01 **GROUP BY** S.Cost, S.Description;	

Results | Messages

	TotalShippingPrice	Description
1	71.67	Freight
2	28.34	Overnight
3	194.70	Two-Day

11	A display of all ski products separated into short (under 150 cm), medium (150 -175 cm), and long (over 175 cm) lengths.	**SELECT** P.ProductID, L.Description **AS** Length,**Case** **When** L.Description <= 150 **Then** 'Short' **When** L.Description <= 175 **Then** 'Medium' **When** L.Description> 175 **Then** 'Long' End **AS** [LengthRange]-- **FROM** TProduct P, TLength L **WHERE** L.LengthID = P.LengthID ;	

Results | Messages

	ProductID	Length	LengthRange
1	1	184	Long
2	2	191	Long
3	3	177	Long
4	4	170	Medium
5	5	177	Long
6	6	184	Long

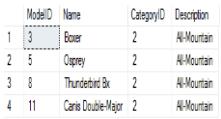
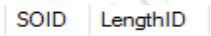
12	List of all customers that have not made a purchase within the last 3 months from the current date.	**SELECT** C.CustomerID, CustFName + ' ' + CustLName **AS** CustomerName, OrderDate **WHERE** C.CustomerID = S.CustomerID **AND** **MONTH**(S.OrderDate) < **DATEADD**(mm, -2, GETDATE())	

	CustomerID	CustomerName	OrderDate
1	1	Andrew Long	2018-05-01
2	2	Mannix Boyd	2019-07-23
3	3	Shelly Woodard	2019-03-20
4	4	Deanna Oneill	2019-09-02
5	5	Arden Thompson	2019-01-04
6	6	Zia Klein	2018-03-25
7	7	Simon Johnson	2019-01-06
8	8	Tobias Estes	2018-05-25

<table>
<tr>
<td>13</td>
<td>List of customers whose average sales is less than the average of all sales. This will
help us to find customers whom we should target to get a higher volume of sales.</td>
<td>SELECT C.CustomerID, CustFName, AVG([SO(OrderTotal)]) AS AvgSales
FROM TCustomer C, TSalesOrder S
WHERE [SO(OrderTotal)] > (SELECT AVG([SO(OrderTotal)]) FROM TSalesOrder)
AND C.CustomerID = S.CustomerID
Group BY C.CustomerID, C.CustFName</td>
<td>
<table>
<tr><th></th><th>CustomerID</th><th>CustFName</th><th>AvgSales</th></tr>
<tr><td>1</td><td>2</td><td>Mannix</td><td>599.66</td></tr>
<tr><td>2</td><td>3</td><td>Shelly</td><td>997.79</td></tr>
<tr><td>3</td><td>4</td><td>Deanna</td><td>634.18</td></tr>
<tr><td>4</td><td>5</td><td>Arden</td><td>690.83</td></tr>
<tr><td>5</td><td>6</td><td>Zia</td><td>687.22</td></tr>
</table>
</td>
</tr>
<tr>
<td>14</td>
<td>The ten most profitable products (i.e., the products that have earned PSC the greatest profits) in a given year.</td>
<td>SELECT TOP 10 SUM(M.MAPPrice - P.Cost) AS [Profit], P.ModelID, P.ProductID
FROM TProduct P, TSalesOrderLine O, TSalesOrder S, TModel M
WHERE M.ModelID = P.ModelID
AND P.ProductID = O.ProductID
AND S.SOID= O.SOID AND YEAR(S.OrderDate) = 2018
GROUP BY P.ModelID, P.ProductID;</td>
<td>
<table>
<tr><th></th><th>Profit</th><th>ModelID</th><th>ProductID</th></tr>
<tr><td>1</td><td>134.87</td><td>4</td><td>1</td></tr>
<tr><td>2</td><td>-6111.01</td><td>4</td><td>6</td></tr>
<tr><td>3</td><td>-5590.01</td><td>2</td><td>8</td></tr>
<tr><td>4</td><td>-4914.01</td><td>7</td><td>10</td></tr>
<tr><td>5</td><td>-9214.01</td><td>5</td><td>14</td></tr>
<tr><td>6</td><td>-6854.01</td><td>5</td><td>17</td></tr>
<tr><td>7</td><td>-7275.01</td><td>9</td><td>18</td></tr>
<tr><td>8</td><td>-8450.01</td><td>5</td><td>20</td></tr>
<tr><td>9</td><td>-5903.01</td><td>6</td><td>25</td></tr>
<tr><td>10</td><td>-6245.01</td><td>6</td><td>26</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>15</td>
<td>The number of distinct products managed by each product manager.</td>
<td>SELECT DISTINCT COUNT(P.PurchOrderID) AS [ProductsManaged], EmpID, EmpFName, EmpLName
FROM TEmployee E, TPurchaseOrder P
WHERE E.EmpID = P.EmpID_Mgr
GROUP BY EmpPosition, EmpID, EmpFName, EmpLName;</td>
<td>
<table>
<tr><th></th><th>ProductsManaged</th><th>EmpID</th><th>EmpFName</th><th>EmpLName</th></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Suki</td><td>Clark</td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>Amethyst</td><td>Mills</td></tr>
<tr><td>3</td><td>1</td><td>3</td><td>Zena</td><td>Montoya</td></tr>
<tr><td>4</td><td>1</td><td>4</td><td>Hyatt</td><td>Shelton</td></tr>
<tr><td>5</td><td>1</td><td>5</td><td>Madaline</td><td>Dudley</td></tr>
</table>
</td>
</tr>
<tr>
<td>16</td>
<td>Defect rate (i.e., number of units rejected after manufacturing) for a given model.</td>
<td>SELECT T1.ProductID,(TotalDefects - 1) AS TotalDefects, TotalCount, ((TotalDefects*1.00/TotalCount*1.00)) AS DefectRate
FROM (SELECT ProductID, SUM(Qty) AS TotalDefects
FROM TSalesOrderLine, TDeliveryIn
WHERE DateReceipt IS NOT NULL
GROUP BY ProductID) AS T1
JOIN (SELECT ProductID, SUM(Qty) AS TotalCount
FROM TSalesOrderLine, TDeliveryIn
WHERE DateReceipt IS NULL
GROUP BY ProductID) AS T2
ON T1.ProductID = T2.ProductID</td>
<td>
<table>
<tr><th>ProductID</th><th>TotalDefects</th><th>TotalCount</th><th>DefectRate</th></tr>
</table>
(Currently no returns in system)
</td>
</tr>
</table>

Three Additional Queries

Query #	Question	Why is this important	SQL	Partial Output	Recap of Findings
1	List only Products belonging to the All-Mountain category	This would be important when wanting to compare sales of each category and also when helping a customer.	**SELECT** ModelID, Name, M.CategoryID, C.Description **FROM** TModel M, TCategory C **WHERE** C.CategoryID = M.CategoryID **AND** C.Description = 'All-Mountain'		The findings were numerous because there are 5/15 of the products listed given to us this makes sense and the All-Mountain sales seem decent.
2	Are there any sales where the sales clerk forgot to enter the Model Length?	This is important in the sale because without this the item will not be made right.	**SELECT** S.SOID, LengthID **FROM** TProduct P, TSalesOrderLine O, TSalesOrder S **WHERE** P.ProductID = O.ProductID **AND** LengthID **IS NULL**;		The findings were noted that no salesperson had forgotten to list the Length, but would be important in future use.
3	Show all Products names for sales that include binding.	Binding is not applied to every sale and is use when customizing the products.	**SELECT** P.ProductID, M.Name, 'yes' **AS** HasBinding **FROM** TProduct P, TBOM BOM, TModel M **WHERE** M.ModelID = P.ProductID **AND** P.ProductID = BOM.ProductID **AND** Binding = '1'		The binding is an important feature in the sale because it is a luxury. That is why we found not may orders with a binding.

User Documentation

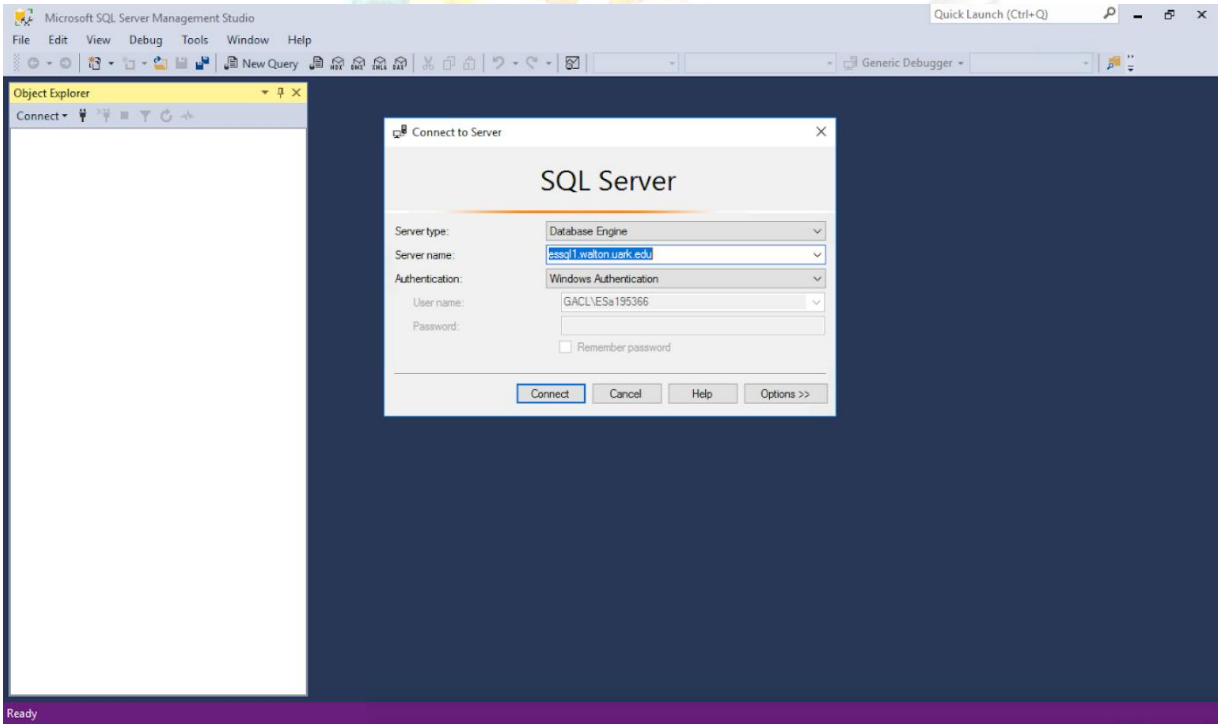
Within the user documentation section are steps on how to access the database. Witin the steps there will be information on queries, and possible issues.

Team SQL Server Account: ESa195366

How to access the database:

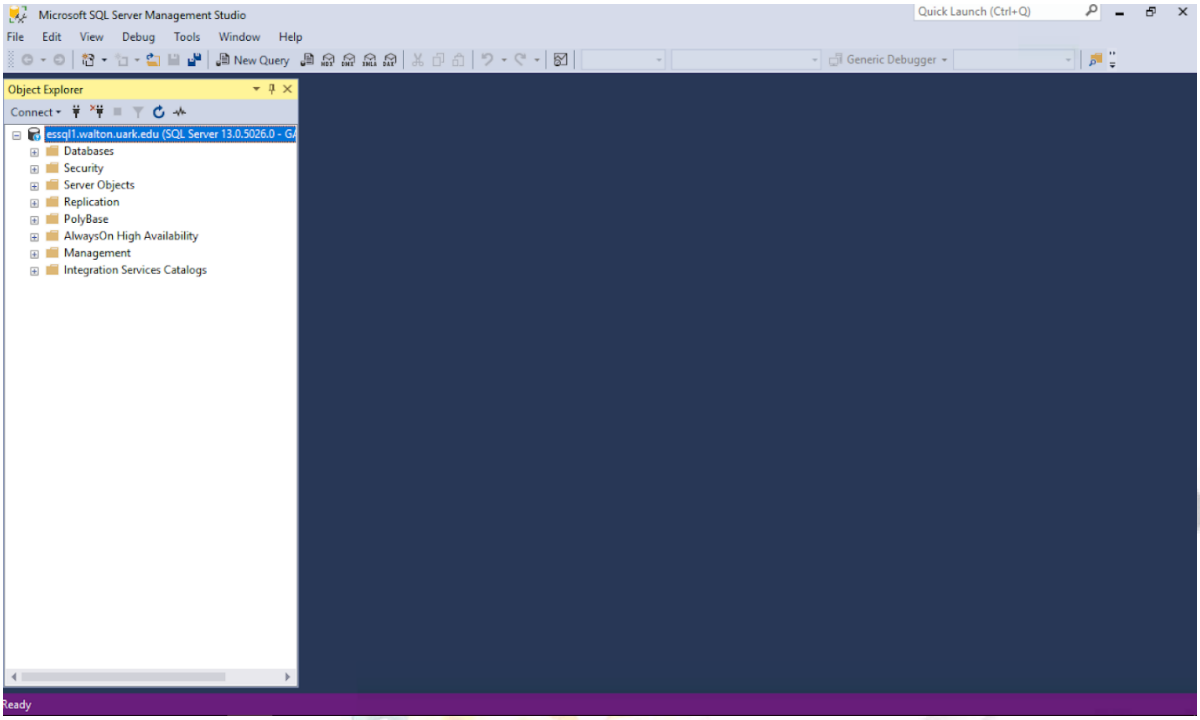
Here we are providing a breakdown of how to navigate the database. This is the desktop icon that we are using:

STEP 1: In order to access this database, you must log onto the SQL Server Management Studio and enter your credentials. We are providing pictures to demonstrate this.



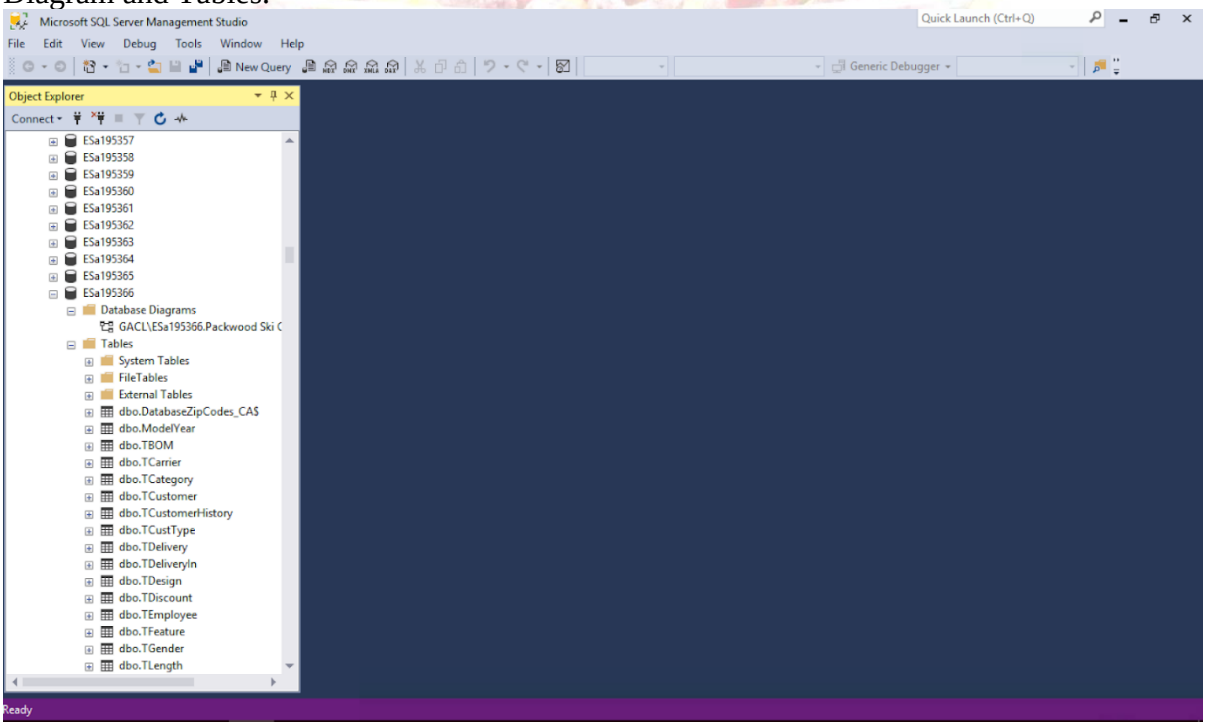
STEP 2: Logging on

If your username is correct and the authentification, your screen should look like the following. If any issues arise, make sure the Authentication says “Windows Authentication”. You will also want to be sure that the server type is “Database Engine.”



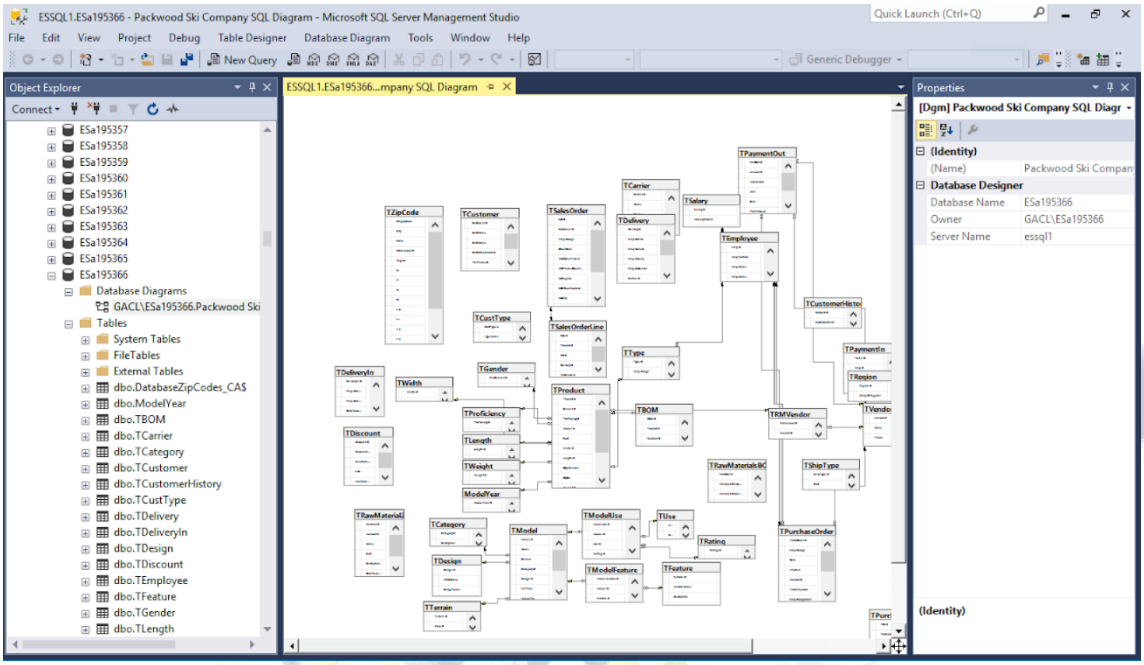
Step 3: Accessing Data

From here, you will need to click the small "plus" sign on the left of the Databases label. This will provide a drop-down of all of the possible databases. For ours, we are using "ESa195366". Here, you will see many different labels. Below shows the dropdown labels for Database Diagram and Tables.



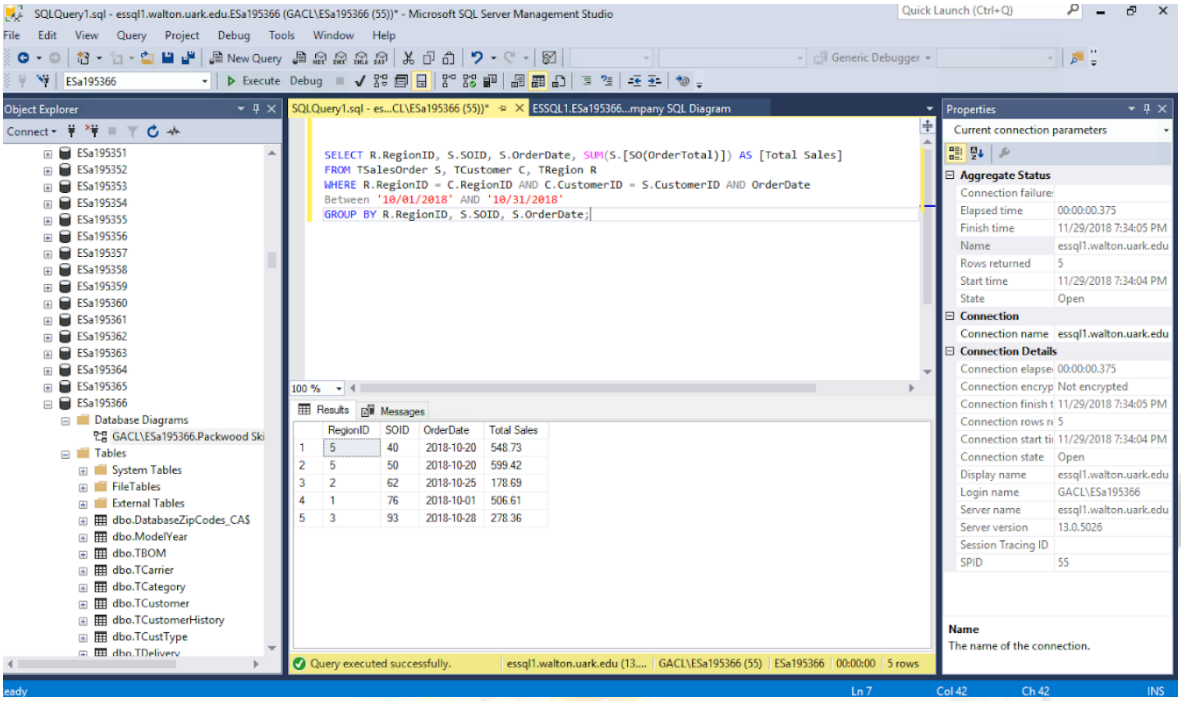
Step 4: Viewing Relationships via Database Diagram

Once you click the “plus sign” next to the Database Diagram label. You will see the diagram that we created for PSC below. Double click on this to view the diagram and its relationships as shown below. This shows all the entities and attributes as well as PK and FK relationships.



Step 5: **Queries**

Queries are running in order to pull up relevant information to make an informed business decision. We are able to obtain multiple different reports by running these queries. First, you will need to click on the “New Query” button located toward the top of the screen. You will see a blank screen where you can then start typing your SELECT, FROM, and WHERE statements. For certain queries, you will also be using ORDER BY, GROUP BY, and HAVING clauses. The query below shows total sales by region in a given month.



Possibles Issues:

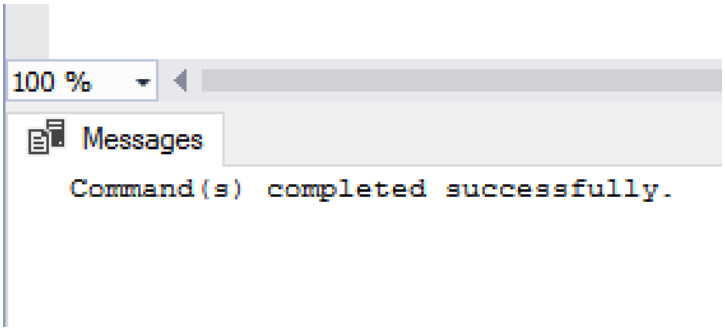
When writing a query, there can be many reasons you might receive an error when trying to run it. You need to double check that you have no spelling errors or mathematical errors. Another issue people run into is incorrect placement of commas or apostrophes.

Step 6: Saving a Query

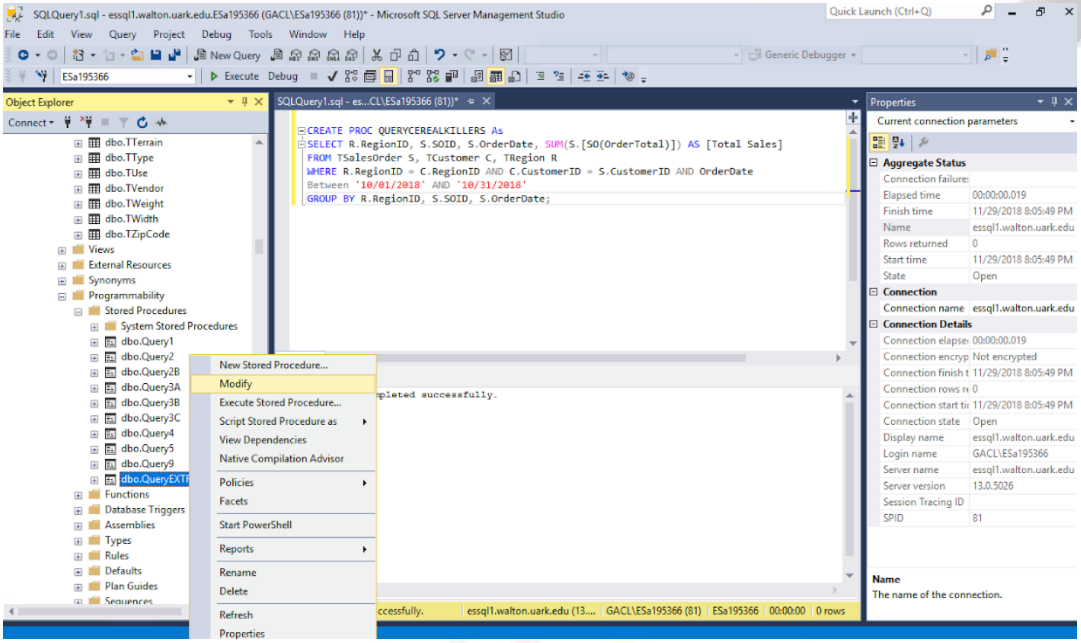
There is one extra line of code you must write in order to save your query. On the first line, enter "CREATE PROC *QueryName* AS". You will need to make sure there are no spaces when deciding the name of your query. Below is a snippet of code to provide better understanding.

```
CREATE PROC QueryEXAMPLE AS
SELECT R.RegionID, S.SOID, S.OrderDate, SUM(
FROM TSalesOrder S, TCustomer C, TRegion R
WHERE R.RegionID = C.RegionID AND C.Customer
Between '10/01/2018' AND '10/31/2018'
GROUP BY R.RegionID, S.SOID, S.OrderDate;
```

Next, you will need to click the "Execute" button. If the query has saved correctly, you will see a message at the bottom of the screen that reads "Command (s) completed successfully."



Step 7:
In order to view/edit your query, scroll down to the "Programmability" label and click the plus sign next to it. Next, click on "Stored Procedures" to view it. This will expand your saved queries. To modify a previously saved query, you can right click on it and click modify in order to change it. Shown below:



What We Learned Throughout This Process

Within this section are paragraphs written by each member. Within their paragraphs they describe what they have individually learned from the Packwood Ski project.

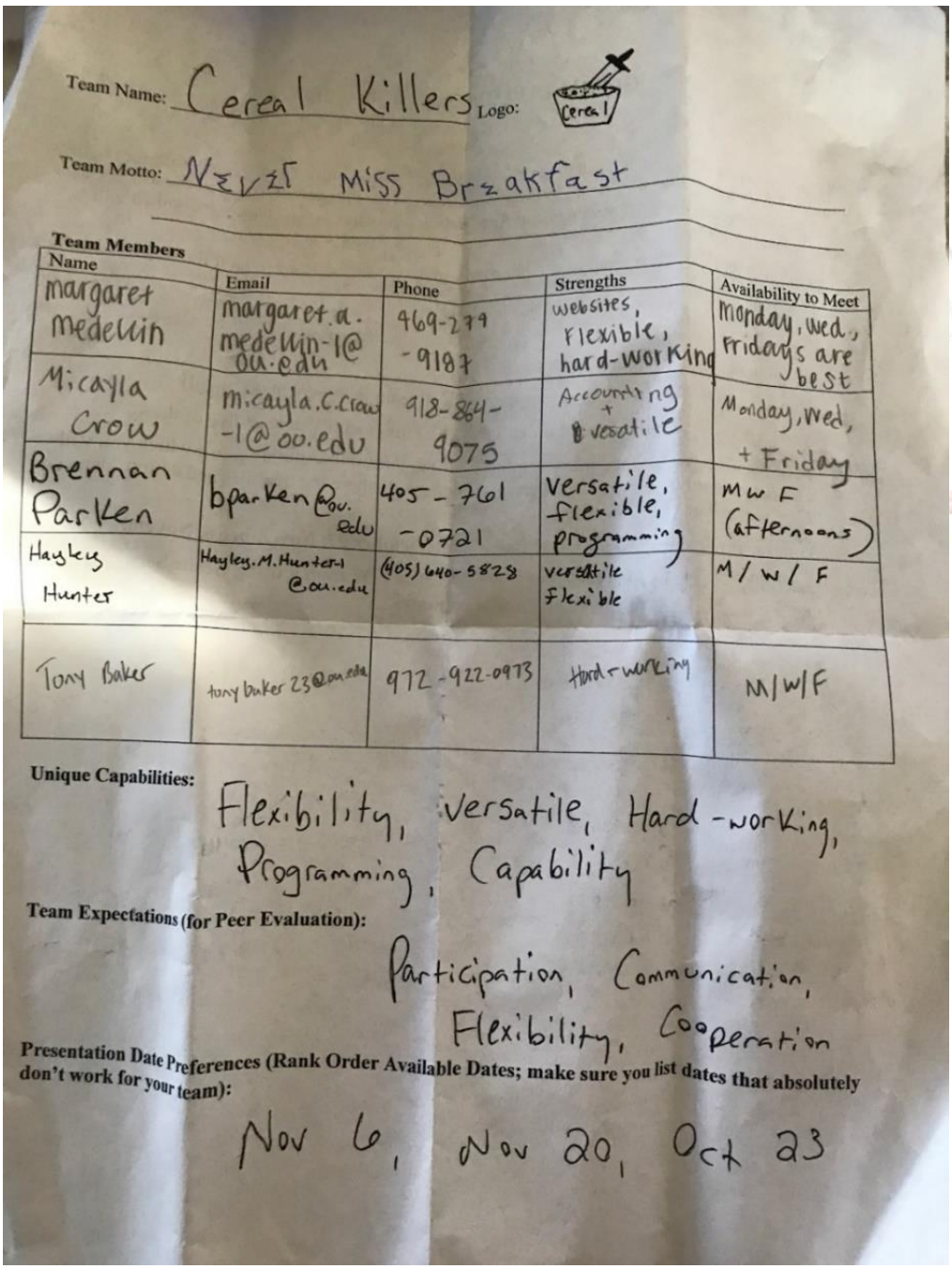
Member Name:	What you learned:
Hayley Hunter	My individual accomplishments involve the understanding of entity relationship diagrams and their relations as well as importance of data integrity. Entity relationship diagrams are important to understand so you can visualize the map of your database system. If I can visualize entities and follow the relations between them, I feel more comfortable in writing the SQL. As I have become more confortable with the tasks thourout the project I found that I worked more and more efficiently and with fewer defects. In addition, integrity is extremely important to me so I was really excited to learn about data integrity. I did not know that there were specific levels of data integrity so when I learned about normal forms zero through three, it was very eye opening.
Margaret Medellin	My individual accomplishments included a better understanding of implementing data into databases and the importance of data relationships in the databases. Before the project, I did not have a good understanding of the implementation of data within a database or the importance of the relationships between entities. Therefore, as the project progressed to Milestone three, I was forced to address many challenges I was not necessary confortable with. However, after completing the final submission I feel as though I have lasting take aways from this project. I am excited to put what I have learned from this project into furture projects.
Micayla Crow	I learned how the importance of the data dictionary as well as how to implement data in a sufficient way. As the project progressed to milestone three, I learned about the innerworkings of the data dictionary model. Often, the model proved challenging to accomplish because I was not familiar with the proper data types to use for some field names. As I addressed my understanding of data types from previous programming languages I have learned throughout my time at the university, I was able to excel in creating the data dictionary model. In addition, I am always looking for a better way to implement data because times is money. So, when I learned of the proper techniques to implement data within the database, I was excited to say the least. I also gained a strong understanding of data relationships as I had to go back and edit data so that the relationships were accurate. As I wrote the queries, I could see first hand how helpful this information is for companies. Being able to pull up relevant data taught me how to make a well-informed decision as a business person in the future.
Brennan Parken	I accomplished learning how to effectively design a database, as well as integrate good integrity and data efficiently. From my experience writing queries, I learned how to effectively use SQL and type effective queries that are efficient and tidy. I also learned how to develop an executive summary for reports. I was able to relate my programming experience and use it with databases, which has helped me understand the impact of data on business.

As a team we learned that about all the processes involved in the creation of a database. As of Milestone 3, we had quite a bit of learning to do, so that we could become comfortable with the development of our database system. A big turning point in our development was the understanding of where we were going wrong while assigning foreign keys to the diagrams. Once we understood how to correctly assign the foreign keys, and became familiar with the SQL server, it was smooth sailing. In addition, we chose to communicat meeting times through text messaging. We also used this source to communicate questions and answers to all team members. Though we had many issues that arose from the implementation of the SQL server, it brought us together as a group. Throughout the project, each of us helped to improve the teams understanding of the tasks at hand. We grew individually and as a team.

Appendix

Below is the team's contract that was filled out by all starting members. The contract holds information for all members so that we know what days work best for everyone as well as the best time for everyone to meet. Most importantly, is the contact information of every team member.

Team Contract:



Team Name: Cereal Killers Logo:

Team Motto: Never Miss Breakfast

Team Members

Name	Email	Phone	Strengths	Availability to Meet
margaret medellin	margaret.a.medellin-1@ou.edu	469-279-9187	websites, flexible, hard-working	Monday, Wed., Fridays are best
Micayla Crow	micayla.c.crow-1@ou.edu	918-864-9075	Accounting + versatile	Monday, Wed, + Friday
Brennan Parken	bparken@ou.edu	405-761-0721	versatile, flexible, programming	MWF (afternoons)
Hayley Hunter	Hayley.M.Hunter-1@ou.edu	(405) 640-5828	versatile flexible	M / W / F
Tony Baker	tonybaker23@ou.edu	972-922-0973	Hard-working	M/W/F

Unique Capabilities:

Flexibility, versatile, Hard-working, Programming, Capability

Team Expectations (for Peer Evaluation):

Participation, Communication, Flexibility, Cooperation

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

Nov 6, Nov 20, Oct 23

Data Dictionary Model

Below are images of our data dictionary model. This model in extremely useful to look at while writing code in SQL. This model serves as a map so that the user knows the data types, field names, and keys of the attributes as well as the tables that they can be found in.

Table	Field Name	Key	Data Type	Null	Default	References	Sample
Customer	CustomerID	PK	Int	Not Null			1
	FName		VarChar(50)	Not Null			Sarah
	LName		VarChar(50)	Not Null			Johnson
	StreetAddress		VarChar(100)	Not Null			Asp Avenue
	CustTypeID	FK	Int	Not Null		CustType	1
	PmtTermsID	FK	Int	Null Allowed		Payment Terms	1
	ZipCodeID	FK	VarChar(10)	Not Null		ZipCode	73069
	RegionID	FK	Int	Not Null		Region	1
CustomerHistory	CustHistID	PK	Int	Not Null			1
	CustHistInfo		VarChar(250)	Not Null			Customer has bought multiple products from us in the past
	CustomerID	FK	Int	Not Null		Customer	1
PaymentTerms	PmtTermsID	PK	Int	Not Null			1
	Length		Int	Not Null			1
	DiscountAmt		Money	Null Allowed			$1.00
	DiscountPeriod		Int	Null Allowed			1
CustType	CustTypeID	PK	Int	Not Null			1
	TypeName		VarChar(50)	Not Null			Retailer
Zipcode	ZipcodeID	PK	Int	Not Null			73069
	CityID	FK	VarChar(50)	Not Null		City	1
	StateID	FK	Char(2)	Not Null		State	TX
	CountryID	FK	VarChar(50)	Not Null		Region	1
	RegionID	FK	VarChar(50)	Not Null		Country	1
Country	CountryID	PK	Int	Not Null			1

Table	Field Name	Key	Data Type	Null	Default	References	Sample
	CountryName		VarChar(50)	Not Null			Canada
State	StateID	PK	Int	Not Null			1
	StateAbbreviation		Char(2)	Not Null			TX
Region	RegionID	PK	Int	Not Null			1
	RegionName		VarChar(50)	Not Null			Midwest
City	CityID	PK	Int	Not Null			1
	CityName		VarChar(50)	Not Null			Dallas
SalesOrder	SOID	PK	Int	Not Null			1
	CustomerID	FK	Int	Not Null		Customer	1
	EmpID_Mgr	FK	Int	Not Null		Employee	1
	OrderDate		Date	Not Null			11/20/17
	[OrderTotal]		Money	Not Null			$500.00
	DiscountID	FK	Int	Null Allowed		Discount	1
	PhoneOrder?(y/n)		TinyInt	Not Null			1
	Region		VarChar(50)	Not Null			Midwest
	StreetAddress		VarChar(100)	Not Null			Asp Avenue
	City		VarChar(50)	Not Null			Dallas
	State		VarChar(50)	Not Null			TX
	ZipCode		VarChar(10)	Not Null			73069
	PmtInID	FK	Int	Not Null		PaymentIn	1
<SalesOrderLine>	SOLID	PK	Int	Not Null			1
	ProductID	FK	Int	Not Null		Product	1
	SOID	FK	Int	Not Null		SalesOrder	1
	DeliveryID	FK	Int	Not Null		Delivery	1
	[SOLTotal]		Money	Not Null			$500.00
	Qty		Int	Not Null			1
	Date		Date	Not Null			11/20/17
Discount	DiscountID	PK	Int	Not Null			1
	DiscountRate		Decimal(4,2)	Not Null			2.5
	RelationLength		Int	Not Null			1
	Size		smallmoney	Not Null			$1.00

Table	Field Name	Key	Data Type	Null	Default	References	Sample
	AdditionalDiscount		smallmoney	Not Null			$1.00
PaymentIn	PmtInID	PK	Int	Not Null			1
	EmpID	FK	Int	Not Null		Employee	1
	PayInDate		Date	Not Null			11/20/17
	Amt		money	Not Null			$1.00
	InvoiceNumber		Int	Not Null			1
	PmtInfo		VarChar(250)	Null Allowed			This payment is to be tracked closely until confirmed
Delivery	DeliveryID	PK	Int	Not Null			1
	EmpID_Pick	FK	Int	Not Null		Employee	1
	EmpID_Pack	FK	Int	Null Allowed		Employee	1
	EmpID_Ship	FK	Int	Null Allowed		Employee	1
	EmpID_Deliver	FK	Int	Not Null		Employee	1
	CarrierID	FK	Int	Not Null		Carrier	1
	ShipDate		Date	Not Null			11/20/17
	ShipStatus		VarChar(30)	Null Allowed			Partial
	ShipTypeID	FK	Int	Not Null		ShipType	1
Carrier	CarrierID	PK	Int	Not Null			1
	Name		VarChar(50)	Not Null			FedEx
	Phone		VarChar(50)	Not Null			(918) 375-6789
	StreetAddress		VarChar(100)	Not Null			Asp Avenue
	City		VarChar(50)	Not Null			Dallas
	State		VarChar(50)	Not Null			TX
	ZipCode		VarChar(10)	Not Null			73069
ShipType	ShipTypeID	PK	Int	Not Null			1
	Cost		Money	Not Null			$1.00
	Description		VarChar(250)	Not Null			Overnight
	Waived?(y/n)		TinyInt	Not Null			1
Salary	SalaryID	PK	Int	Not Null			1
	[SalaryTotal]		Money	Not Null			$500.00
Region	RegionID	PK	Int	Not Null			1

Table	Field Name	Key	Data Type	Null	Default	References	Sample
	EmpID_Region	FK	Int	Not Null		Employee	1
	Description		VarChar(250)	Not Null			Midwest
Employee	EmpID	PK	Int	Not Null			1
	EmpPosition		VarChar(250)	Null Allowed			Manager
	EmpFName		VarChar(50)	Not Null			John
	EmpLName		VarChar(50)	Not Null			Mayer
	Commission?(y/n)		TinyInt	Not Null			1
	SalaryID	FK	Int	Not Null		Salary	1
DeliveryIn	DeliveryInID	PK	Int	Not Null			1
	EmpID_received	FK	Int	Not Null		Employee	1
	EmpID_inspects	FK	Int	Not Null		Employee	1
	DateReceipt		Date	Not Null			11/20/17
PurchaseOrder	PurchOrderID	PK	Int	Not Null			1
	EmpID_Mgr	FK	Int	Not Null		Employee	1
	Date		Date	Not Null			11/20/17
	[Total]		Money	Not Null			$500.00
	VendorID	FK	Int	Not Null		Vendor	1
	PurchaseReqNum		Int	Not Null			1
	EmpID_approves	FK	Int	Not Null		Employee	1
	DateApproved		Date	Not Null			11/20/17
	EmpID_create	FK	Int	Not Null		Employee	1
	DateCreated		Date	Not Null			11/20/17
<PurchaseOrderLine>	POLID	PK	Int	Not Null			1
	PurchOrderID	FK	Int	Not Null		PurchaseOrder	1
	ProductID	FK	Int	Not Null		Product	1
	DeliveryInID	FK	Int	Null Allowed		DeliveryIn	1
	Qty		TinyInt	Not Null			1
	[LineTotal]		Money	Not Null			$500.00
PaymentOut	PmtOutID	PK	Int	Not Null			1
	VendorID	FK	Int	Not Null		Vendor	1
	InvoiceNum		Int	Not Null			1
	Amt		smallmoney	Not Null			$1.00

Table	Field Name	Key	Data Type	Null	Default	References	Sample
	Date		Date	Not Null			11/20/17
	PmtInfo(…)		VarChar(250)	Null Allowed			This payment is to be tracked closely until confirmed
	PurchOrderID	FK	Int	Not Null		PurchaseOrder	1
Type	TypeID	PK	Int	Not Null			1
	EmpID_Mgr	FK	Int	Not Null		Employee	1
	TypeName		VarChar(50)	Not Null			Racing
Vendor	VendorID	PK	Int	Not Null			1
	Name		VarChar(50)	Not Null			Adams
	Phone(…)		VarChar(10)	Not Null			(918) 375-6789
	ContactName		VarChar(50)	Not Null			Adams
	Address		VarChar(100)	Not Null			Asp Avenue
<RawMatVendor>	RMVendorID	PK	Int	Not Null			1
	VendorID	FK	Int	Not Null		Vendor	1
	RawMatID	FK	Int	Not Null		RawMaterials	1
	Price		smallmoney	Not Null			$1.00
RawMaterials	RawMatID	PK	Int	Not Null			1
	VendorID	FK	Int	Not Null		Vendor	1
	Name		VarChar(50)	Not Null			Fiberglass
	DateReceived		Date	Not Null			11/20/17
	Description		VarChar(250)	Null Allowed			Must be cut to proper shape
	Cost		smallmoney	Not Null			$1.00
<RawMaterialsBOM>	RMBOMID	PK	Int	Not Null			1
	RawMatId_assembly	FK	Int	Not Null		RawMaterials	1
	RawMat_component	FK	Int	Not Null		RawMaterials	1
	Qty		TintInt	Not Null			1
<BillOfMaterials>	BOMID	PK	Int	Not Null			1
	ProductID	FK	Int	Not Null		Product	1
	Qty		Int	Not Null			1
	RawMatID	FK	Int	Not Null		RawMaterials	1
	Binding?(y/n)		TintInt	Not Null			1

Table	Field Name	Key	Data Type	Null	Default	References	Sample
Product	ProductID	PK	Int	Not Null			1
	GenderID	FK	Int	Not Null		Gender	1
	SizeID	FK	Int	Not Null		Size	1
	ProficiencyID	FK	Int	Not Null		Proficiency	1
	ModelID	FK	Int	Not Null		Model	1
	[Cost]		money	Not Null			$1.00
	TypeID	FK	Int	Not Null		Type	1
	WidthID	FK	Int	Not Null		Width	1
	LengthID	FK	Int	Not Null		Length	1
	[QtyAvailable]		Int	Null Allowed			1
	QtyOH		Int	Not Null			1
	WeightID	FK	Int	Not Null		Weight	1
	ModelYearID	FK	Int	Not Null		ModelYear	1
	RMVendorID	FK	Int	Not Null		<RawMatVendor>	1
Length	LengthID	PK	Int	Not Null			1
	Description		VarChar(250)	Not Null			171,179,187,195
Gender	RM_genderID	PK	Int	Not Null			1
	Name		VarChar(50)	Not Null			Female
Proficiency	ProficiencyID	PK	Int	Not Null			1
	Name		VarChar(50)	Not Null			Beginner
Width	WidthID	PK	Int	Not Null			1
	Description		VarChar(250)	Not Null			133x112x123 (195)
ModelYear	ModelYearID	PK	Int	Not Null			1
	Description		VarChar(50)	Not Null			2016
Weight	WeightID	PK	Int	Not Null			1
	Description		VarChar(250)	Not Null			2.1 kg (195)
Model	ModelID	PK	Int	Not Null			1
	Name		VarChar(50)	Not Null			Eagle
	Gender		VarChar(10)	Not Null			M/F
	CategoryID	FK	Int	Not Null		Category	1
	DesignID	FK	Int	Not Null		Design	1
	MAPEnd		Date	Not Null			11/20/17

Table	Field Name	Key	Data Type	Null	Default	References	Sample
	ImageFile		VarChar(250)	Null Allowed			Raven2016.jpg
	MAPPrice		Money	Null Allowed			$500.00
	Description		VarChar(250)	Null Allowed			Our latest and lightest design
	TerrainID	FK	Int	Not Null		Terrain	1
<ModelUse>	ModelUseID	PK	Int	Not Null			1
	ModelID	FK	Int	Not Null		Model	1
	UseID	FK	Int	Not Null		Use	1
	RatingID	FK	Int	Not Null		Rating	1
	Note		VarChar(250)	Null Allowed			ModelforEasternSports,Inc.only
Use	UseID	PK	Int	Not Null			1
	Description		VarChar(250)	Null Allowed			Touring
Rating	RatingID	PK	Int	Not Null			1
	Description		VarChar(250)	Null Allowed			I loved my ski
Design	DesignID	PK	Int	Not Null			1
	ArtistName		VarChar(50)	Not Null			Shotridge
	DesignName		VarChar(50)	Not Null			Rainbow
Terrain	TerrainID	PK	Int	Not Null			1
	EmpID	FK	Int	Not Null		Employee	1
	Description		VarChar(250)	Not Null			All-Mountain
Category	CategoryID	PK	Int	Not Null			1
	Description		VarChar(250)	Not Null			Rental
<ModelFeature>	ModelFeatureID	PK	Int	Not Null			1
	ModelID	FK	Int	Not Null		Model	1
	FeatureID	FK	Int	Not Null		Feature	1
Feature	FeatureID	PK	Int	Not Null			1
	FeatureName		VarChar(50)	Not Null			Anti-Vibration
	Description		VarChar(250)	Null Allowed			Prevents skier from feeling vibrations

Project Management

With using the projectMIS tool to keep track, we have learned that there is a lot that goes into managing a group project, especially of this scale. It has helped us see and manage tasks as well as team members participation. Often, the team faced challenges with the management tool (ProjectMIS). However, as the project came to a close the management tool became easier to understand and therefore, we spent less time trying to figureout how/where we should be logging in our hours spent on the project. In addition, the the hours spent we discovered how important the hourly rate was with our baseline budgeting. We had to address our budget and work toether to find possible solutions for our lack of rescources.

Milestone 1

CPI is computed by Earned Value / Actual Cost. A value of above 1 means that the **project** is doing well against the budget. The Earned Value for the client meeting and milestone 1 has a total of $216. The Actual Cost for the client meeting and milestone 1 has a total of $104. Our computed CPI is 2.08, which is very high performance with fairly low costs.

SPI is computed by Earned Value / Planned Value. **SPI** represents how close actual work is being completed compared to the schedule. The Earned value is $216 and the planned value is $168. The calculated SPI is 1.29, which is still a high-performance index and means we planned for a decent number of budgeted hours to work.

Milestone 2

CPI is computed by Earned Value / Actual Cost. A value of above 1 means that the **project** is doing well against the budget. The Earned Value is $400. The Actual Cost has a total of $400. Our computed CPI is 1, which is good.

SPI is computed by Earned Value / Planned Value. **SPI** represents how close actual work is being completed compared to the schedule. The Earned value is $400 and the planned value is $400. The calculated SPI is 1. **Earned Value Analysis:** compares the performance measurement base to the actual schedule & cost performance.

Milestone 3

CPI is computed by Earned Value / Actual Cost. A value of above 1 means that the **project** is doing well against the budget. The Earned Value for the client meeting and milestone 3 has a total of $160. The Actual Cost for the client meeting and milestone 3 has a total of $520. Our computed CPI is .31, which is very high performance with fairly low costs.

SPI is computed by Earned Value / Planned Value. **SPI** represents how close actual work is being completed compared to the schedule. The Earned value is $160 and the planned value is $126.4. The calculated SPI is 1.27, which is still a high-performance index and means we planned for a decent number of budgeted hours to work. **Earned Value Analysis:** compares the performance measurement base to the actual schedule & cost performance.

Final Submission

CPI is computed by Earned Value / Actual Cost. A value of above 1 means that the **project** is doing well against the budget. The Earned Value is $1,016. The Actual Cost is $1,484. Our computed CPI is .68, which is very high performance with fairly low costs.

SPI is computed by Earned Value / Planned Value. **SPI** represents how close actual work is being completed compared to the schedule. The Earned value is $1,016 and the planned value is $972.80. The calculated SPI is 1.04, which is a good performance index and means we planned well. **Earned Value Analysis:** compares the performance measurement base to the actual schedule & cost performance.

- **Planned Value:** authorized budget to schedule work.
- **Earned Value:** measure at the work performance expressed in terms of the budget authorized to that work.
- **Actual Cost:** realized cost incurred for the work performance on an activity during a specific time period.
- **Budgeted amount:**
  - Hours x time worth x people = collective budgeted amount
  - PV = percent planned to complete x budgeted amount
  - EV= percent complete x budgeted amount
  - SPI=EV/PV
  - CPI = EV/AC

Project MIS Packwood Ski Company

Project Dashboard Initiating Planning Executing Monitoring and Controlling Help with this Page

Project Dashboard

Current Baselines: $1,016.00 2018-12-03

1 Packwood Ski Company	Planned Start	Planned Finish	Planned Cost	Actual Start	Actual Finish	Actual Cost
1.1 Client Meeting	2018-09-25	2018-10-15	$56.00	2018-10-01	2018-12-02	$24.00
1.2 Milestone 1	2018-10-16	2018-10-25	$160.00	2018-10-01	2018-12-02	$80.00
1.3 Milestone 2	2018-10-26	2018-11-08	$400.00	2018-10-21	2018-12-02	$400.00
1.4 Milestone 3	2018-11-09	2018-11-22	$160.00	2018-11-09	2018-12-02	$920.00
1.5 Final Project	2018-11-23	2018-12-03	$240.00	2018-12-02	2018-12-02	$60.00

1 Packwood Ski Company	Percent Complete	Earned Value	Actual Cost	Planned Value	Cost Variance	Schedule Variance	Cost Performance Index	Schedule Performance Index
1.1 Client Meeting	1.00	$56.00	$24.00	$56.00	$32.00	$0.00	2.33	1.00
1.2 Milestone 1	1.00	$160.00	$80.00	$160.00	$80.00	$0.00	2.00	1.00
1.3 Milestone 2	1.00	$400.00	$400.00	$400.00	$0.00	$0.00	1.00	1.00
1.4 Milestone 3	1.00	$160.00	$920.00	$160.00	-$760.00	$0.00	0.17	1.00
1.5 Final Project	1.00	$240.00	$60.00	$196.80	$180.00	$43.20	4.00	1.22
Totals	1.00	$1,016.00	$1,484.00	$972.80	-$468.00	$43.20	0.68	1.04