

University of Oklahoma

Space Jam
Canon Frye, Reid Harrington
MIS 3353 – Database Management
Dr. Bellah
10/24/18

Executive Summary

This report provides details, diagrams and research about how we have handled this project and created a strong database to better enable Packwood to store information and how to use our database. Our team Space Jam researched the Packwood Ski Company to allow us to create the best possible database.

Space Jam has created this strong database through its dedicated members and effective communication. Space Jam is a partnership that thrives on planning and our timely manner of completing tasks and splitting up the work by each member's special skills. Space Jam is effective because we hold ourselves to a high standard and always motivate each member to perform at their best.

The database we have created was designed to unify the recording and reporting of the processes of Packwood Ski Company. To do this we had to ask questions about the specific information Packwood wanted to track, its operations and its products. We created an entity relationship diagram with some assumptions to model our database. By normalizing our ERD we were able to review its referential integrity and strengthen our database and rid it of unnecessary redundancies. By denormalizing our ERD we allowed certain redundancies to improve our database. Finally, we implemented the design by creating each table in SQL Server Management Studio through a series of SQL Statements.


Once created, each table was populated with 10 "dummy data" entries so we could test the effectiveness of the database. After each table was populated, we could then run SQL queries to find the data requested by PSC. Everything is shown and introduced below.

In the end, we calculated our CPI and SPI resulting in astounding numbers for our CPI. This shows how effective we were with our time management and speed with getting the required tasks complete.

Contents

Executive Summary	2
Get to Know the Team: (Space Jam)	4
Conceptual Design	5
The Client Meeting	5
Q&A During the Meeting & Information We Learned	5
Significant Assumptions	6
What is an ERD? Why is it necessary?	6
Business Cycles Used	6
ERD Created	7
Changes made to generic ERDs	8
Logical Design	9
Normalization	9
Normalized Relations	9
Differences between ERD and Normalized Relations	11
Referential Integrity	11
Physical Design and Implementation	12
Data Dictionary	12
Denormalization	12
Implemented Physical Design	12
Challenges Faced/Addressed During Implementation	13
Strengths and Weaknesses Encountered During Implementation	13
Specific SQL Statements Requested	13
Three Additional Queries	17
User Documentation	19
What We Learned Throughout This Process	21
Appendix	22
Team Contract	22
Data Dictionary Model	22
Project Management	25

Get to Know the Team: (Space Jam)

	
Canon Frye	Reid Harrington
MIS and Supply Chain	MIS
Junior	Junior
From Fort Worth, Texas	From Little Rock, Arkansas

Space Jam was formed in 2018 as a Database Management group with a focus on helping businesses maximize efficiency in product production, sales, and purchasing. We have consolidated our team to maximize our personal efficiency as well. The two of us are very strong in our fields and have figured out how to manage the workflow to our high standards. We both have a background in MIS and as we begin to take on more contracts, we can only learn more and more.

Conceptual Design

The Client Meeting

The client meeting is where we met as a group with our client to clarify any questions we had after reading through the client's request. These questions were mostly to clarify assumptions we would later be making when creating the ERD, but some were used to figure out what entities were to be used, and to get the attributes for some of the larger entities. Some requests in the document were vague and the client meeting allowed us to find out exactly what the client wanted us to do.

- Meeting Time: 12:30pm Thursday Oct. 4, 2018
- Location: Adams Hall room 4
- Interviewers: Reid Harrington, Canon Frye, Kenna Broomfield
- Interviewee: Dr. Jeremy Bellah

Q&A During the Meeting & Information We Learned

What information is needed to track growth?

Revenue, Raw Materials Inventory, COGS, Profit Margin

What is required in a purchase document?

Employee, Date Ordered, Items, Quantity, Date Received, Inspector, Date Inspected

What are the current operating departments?

Production, Sales, Purchasing, Shipping, Marketing

Are the different lengths of skis comprehensive, or are there any special cases?

There are no special cases, all are comprehensive.

Do you keep track of all your retailers in one place, or do you separate large and small ones?

They are all kept in one place.

Significant Assumptions

Assumptions make up a great deal of ERDs, as we need to make them for a lot of our relationship choices. Assumptions may be cleared up by asking very specific questions of the person requesting the ERD, but to make every single assumption clear would take much more work and probably cost more of the person/company.

Five significant assumptions we made are as follows:

- By using IsOpenInvoice in Customer, we can tell if they do not want an open invoice, they want a balance forward invoice.
- All different product categories are covered by our four reference tables.
- Not all of our raw materials may come from vendors, so we have a zero to many relationship.
- Even though we use third party carriers, we still pick, pack, and get our products shipped.
- Some raw materials are used to make WIP components which are then used as raw materials themselves. This is assumed to be covered by the <RMBOM> entity.

What is an ERD? Why is it necessary?

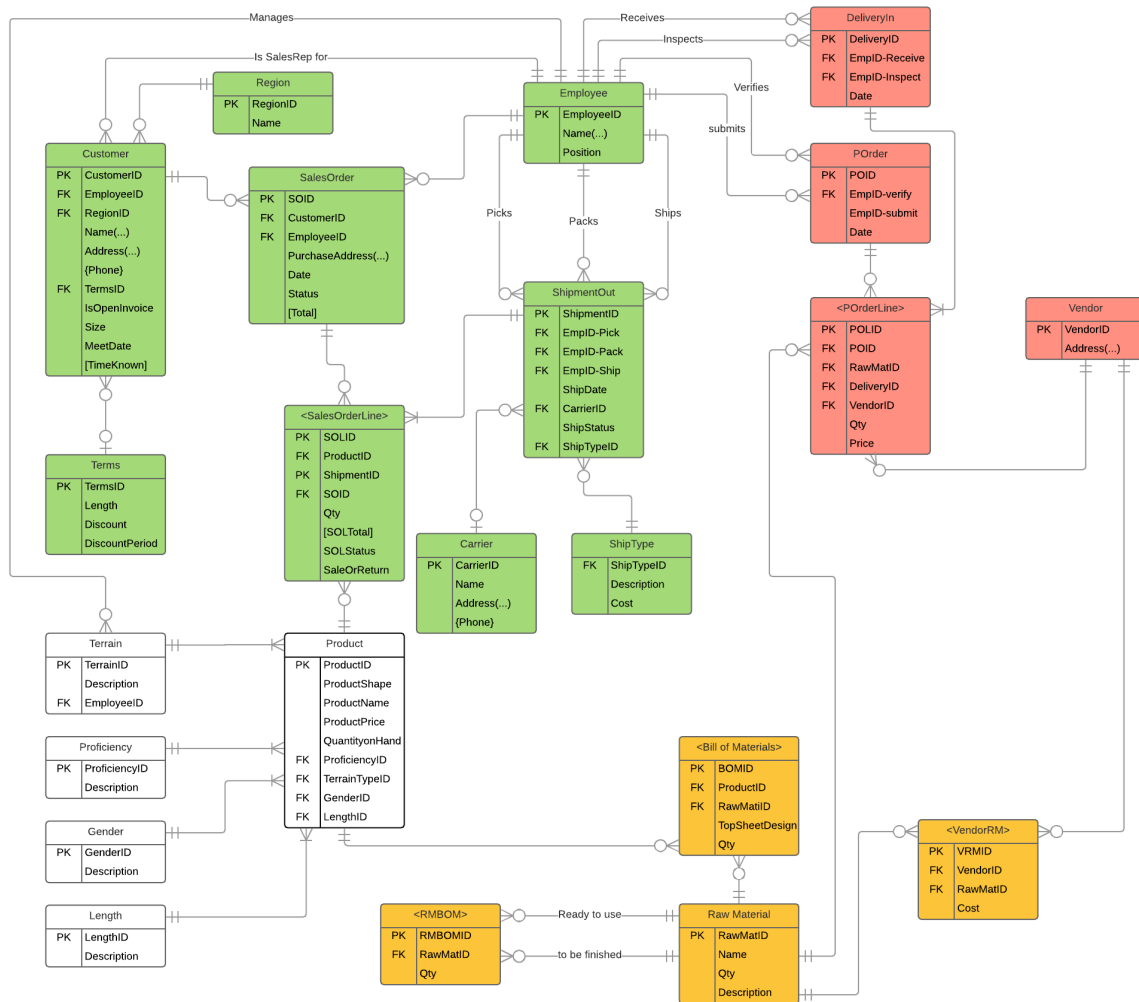
The Entity Relationship Diagram (ERD) is the graphical database model, it represents information and relationships between entities and instances that is stored in the database. Creating ERD is important because it is the visual design of the information stored in the database and from the ERD we can structure queries to generate meaningful information. In our project, we created ERD for Packwood Ski business cycles production, expenditure, and revenue. All data within the ERD represents valuable information for different functions of the company like the names and contact information for Packwood Ski customers are stored in one entity and it connects with sales information in another entity.

Business Cycles Used

In our ERD we used the production cycle, expenditure cycle, and revenue cycle. We used the production cycle because Packwood ski company manufacturers skii's of different raw materials. We used the expenditure cycle because Packwood ski company purchases bindings and other manufacturing supplies from third parties so we have to track our purchase orders, deliveries and from what vendor. We used the revenue cycle because Packwood ski company sells their skii's to retailers and wholesalers so we have to track customers, sales orders, employees and shipments.

ERD Created

The ERD we created combines all three business cycles into one complete ERD. This ERD allows us to visualize what the physical database will look like in the future. We can also show and explain the meaning of our specific ERD to our client to make sure it covers everything they wish before going into actual logic design.



Changes made to generic ERDs

Change #	Original ERD	Updated ERD
<p>Revenue Cycle changes: We added a ShipType Entity to account for when we do overnight or special shipping and how much they cost. We also had several more attributes in the Customer entity to cover things such as the amount of time we've worked with the company. We also connected Employee to Customer to show that each customer has a dedicated sales rep.</p>		
<p>Expenditure Cycle changes: Per request of the company, we removed Purchase Requisitions. Also per request of the business, we have 2 employeeIDs in the POrder table to keep track of who originally requested the order, and who approved it</p>		
<p>Production Cycle changes: We included more attributes in the Bill of Materials to account for extra things the company wanted to track.</p>		

Logical Design

Logical design is where we start to move away from an almost purely visual standpoint like an ERD to a more practical point with normalized relations. By making normalized relations based on our ERD, we are able to make sure our database will run as efficiently as possible.

Normalization

Normalization is organizing database fields and tables to minimize redundancy and strengthen our ERD. Normalization is important to include in this project because it will help us see where our ERD is flawed with data redundancy and how we can clean up and strengthen our ERD so that it doesn't experience issues.

Normalized Relations

T_RMBOM(RMBOMID, RawMatID, Qty)

FK RawMatID references T_RawMaterial on delete restrict

T_RawMaterial(RawMatID, Name, Qty, Description)

T_BillOfMaterials(BOMID, ProductID, RawMatID, TopSheetDesign, Qty)

FK ProductID references T_Product on delete restrict

FK RawMatID references T_RawMaterial on delete restrict

T_VendorRM(VRMID, VendorID, RawMatID, Cost)

FK VendorID references T_Vendor on delete restrict

FK RawMatID references T_RawMaterial on delete restrict

T_Vendor(VendorID, StreetName, StreetNumber, State, City, ZipCode)

T_POrderLine(POLID, POID, VPID, RawMatID, DeliveryID, Qty, Price)

FK POID references T_POrder on delete restrict

FK RawMatID references T_RawMaterial on delete restrict

FK DeliveryID references T_DeliveryIn on delete restrict

T_POrder(POID, EmpID-verify, EmpID-submit, date)

FK EmpID-verify references T_Employee on delete restrict

FK EmpID-submit references T_Employee on delete restrict

T_DeliveryIn(DeliveryID, EmpID-receive, EmpID-Inspect, Date)

FK EmpID-receive references T_Employee on delete restrict

FK EmpID-inspect references T_Employee on delete restrict

T_Customer(CustomerID, EmployeeID, RegionID, TermsID, FirstName, LastName, Street, StreetNumber, State, City, ZipCode, Phone, IsOpenInvoice, Size, MeetDate)

FK EmployeeID references T_Employee on delete restrict

FK RegionID references T_Region on delete restrict

FK TermsID references T_Terms on delete set null

T_Region(RegionID, Name)

T_Terms(TermsID, Length, Discount, DiscountPeriod)

T_SalesOrder(SOID, CustomerID, EmployeeID, Street, StreetNumber, City, State, ZipCode, Date, Status)

FK CustomerID references T_Customer on delete restrict

FK EmployeeID references T_Employee on delete restrict

T_SalesOrderLine(SOLID, ShipmentID, ProductID, SOID, Qty, SOLStatus, SaleOrReturn)

FK ShipmentID references T_ShipmentOut on delete restrict

FK ProductID references T_Product on delete restrict

T_Employee(EmployeeID, FirstName, LastName, Position)

T_ShipmentOut(ShipmentID, EmpID-Pick, EmpID-Pack, EmpID-Ship, CarrierID, ShipTypeID, ShipDate, ShipStatus)

FK EmpID-Pick references T_Employee on delete restrict

FK EmpID-Pack references T_Employee on delete restrict

FK EmpID-Shil references T_Employee on delete restrict

FK CarrierID references T_Carrier on delete set null

FK ShipTypeID references T_ShipType on delete restrict

T_Carrier(CarrierID, Name, Street, StreetNumber, City, State, ZipCode, Phone)

T_ShipType(ShipTypeID, Description, Cost)

T_Product(ProductID, ProficiencyID, TerrainTypeID, GenderID, LengthID, ProductShape, ProductName, ProductPrice, QuantityOnHand)

FK ProficiencyID references T_Proficiency on delete restrict

FK TerrainTypeID references T_Terrain on delete restrict

FK GenderID references T_Gender on delete restrict

FK LengthID references T_Length on delete restrict

T_Terrain(TerrainID, EmployeeID, Description)

FK EmployeeID references T_Employee on delete restrict

T_Proficiency(ProficiencyID, Description)

T_Gender(GenderID, Description)

T_Length(LengthID, Description)

Differences between ERD and Normalized Relations

The main difference between the ERD and Normalized Relations is how it's displayed, and how it shows how foreign keys are related to their respective primary keys. In the Normalized Relations, we are able to see phrases like "on delete restrict" or "on delete set null," and these allow us as we develop the database to determine what happens if a data value for a primary key is removed, how it affects the foreign keys referencing it. Another key difference is multi-valued attributes. Where something may have previously shown as "Address(...)," it is now broken into each of its parts. The same attribute can be seen as "Street, StreetNumber, City, State." These differences help us move towards a more usable model instead of a more visual model.

Referential Integrity

Referential integrity is a concept that states that if a foreign key in a field references a primary key in another field it must be the same. Also if a primary key is changed then all of the foreign keys must change as well to stay consistent with the primary key. Referential integrity is important because it helps rid the database of errors. We also discovered errors in our database by normalizing it so we removed unnecessary fields.

Physical Design and Implementation

Physical Design is where we move to the fully practical application. This is the actual database that we can run queries and find the data we need. The implementation required running many lines of code to create the tables outlined by our ERD. This time around, the ERD is smaller as it has been “denormalized,” which is defined below. We also created a data dictionary to help outline the data types for each of our tables.

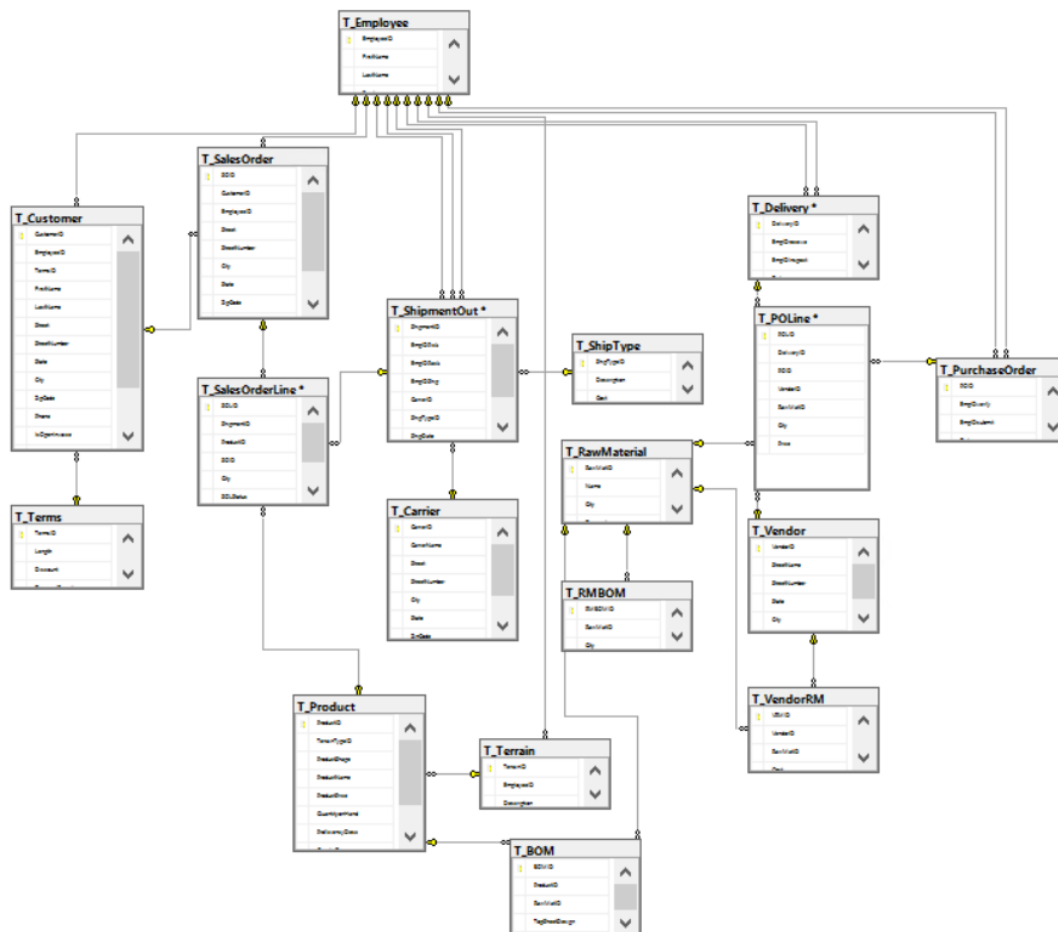
Data Dictionary

A data dictionary is a table that stores information about the contents of a database. The data dictionary holds information about the fields’ data type, size, if it is null, its references, its default and the table it belongs to. It is important because it shows errors and omissions in the database.

Denormalization

Denormalization is an attempt to improve the performance of a database by allowing certain redundancies. Denormalization includes combining two relations into a single relation. We used denormalization in our ERD by adding the reference tables of Terrain, Proficiency, Gender and Length to the Product table. We also used denormalization to add the Reference table Region to the customer table. This change will improve our database performance.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

One major challenge was using the SQL Server Management Studio interface to generate tables took a very long time. To make life easier, I used modified versions of our normalized relations in CREATE TABLE statements I could copy/paste to make the process much faster.

Another challenge came when modifying the normal relations. As they were, there were no data types assigned to them, but through using the data dictionary, it was easier to go through and add the data types to the relations.

Strengths and Weaknesses Encountered During Implementation

Strengths: Organization, familiarity with ERDs, good communication.

Weaknesses: No experience with SSMS, Struggle with importing data.

Specific SQL Statements Requested

PSC asked for a few specific things to be found using our “dummy data” along with some data from a product list provided by the company. Each question listed below is followed by a SQL statement and a screenshot of the results from running the statement. By running these statements, we are able to show that the database we created and gather valuable data for PSC such as sales information to help track the growth of the company.

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).	<pre>SELECT Region, FORMAT(SUM(Qty * ProductPrice), 'C', 'en-us') AS Total, MONTH(Date) AS Month FROM T_Customer C, T_SalesOrder SO, T_SalesOrderLine SOL, T_Product P WHERE C.CustomerID = SO.CustomerID AND SO.SOID = SOL.SOID AND SOL.ProductID = P.ProductID AND MONTH(Date) = 10 GROUP BY Date, Region</pre>	<table><thead><tr><th></th><th>Region</th><th>Total</th><th>Month</th></tr></thead><tbody><tr><td>1</td><td>US</td><td>\$15,874.75</td><td>10</td></tr><tr><td>2</td><td>UK</td><td>\$24,524.65</td><td>10</td></tr></tbody></table>		Region	Total	Month	1	US	\$15,874.75	10	2	UK	\$24,524.65	10																														
	Region	Total	Month																																										
1	US	\$15,874.75	10																																										
2	UK	\$24,524.65	10																																										
2	Total sales (in dollars) by customer in a given year.	<pre>SELECT DISTINCT C.CustomerID, FirstName, LastName, FORMAT(SUM(Qty * ProductPrice), 'C', 'en-us') AS Total, YEAR(Date) AS Year FROM T_Customer C, T_SalesOrder SO, T_SalesOrderLine SOL, T_Product P WHERE C.CustomerID = SO.CustomerID AND SO.SOID = SOL.SOID AND SOL.ProductID = P.ProductID AND YEAR(Date) = 2018 GROUP BY C.CustomerID, FirstName, LastName, YEAR(Date)</pre>	<table><thead><tr><th></th><th>CustomerID</th><th>FirstName</th><th>LastName</th><th>Total</th><th>Year</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Fred</td><td>Jones</td><td>\$52,499.30</td><td>2018</td></tr><tr><td>2</td><td>3</td><td>Matthew</td><td>Smith</td><td>\$6,999.90</td><td>2018</td></tr><tr><td>3</td><td>4</td><td>Frank</td><td>Ocean</td><td>\$75,498.85</td><td>2018</td></tr><tr><td>4</td><td>6</td><td>Sarah</td><td>Franz</td><td>\$24,524.65</td><td>2018</td></tr><tr><td>5</td><td>7</td><td>Kevin</td><td>Brown</td><td>\$71,399.00</td><td>2018</td></tr><tr><td>6</td><td>8</td><td>Mary</td><td>Bell</td><td>\$15,874.75</td><td>2018</td></tr></tbody></table>		CustomerID	FirstName	LastName	Total	Year	1	1	Fred	Jones	\$52,499.30	2018	2	3	Matthew	Smith	\$6,999.90	2018	3	4	Frank	Ocean	\$75,498.85	2018	4	6	Sarah	Franz	\$24,524.65	2018	5	7	Kevin	Brown	\$71,399.00	2018	6	8	Mary	Bell	\$15,874.75	2018
	CustomerID	FirstName	LastName	Total	Year																																								
1	1	Fred	Jones	\$52,499.30	2018																																								
2	3	Matthew	Smith	\$6,999.90	2018																																								
3	4	Frank	Ocean	\$75,498.85	2018																																								
4	6	Sarah	Franz	\$24,524.65	2018																																								
5	7	Kevin	Brown	\$71,399.00	2018																																								
6	8	Mary	Bell	\$15,874.75	2018																																								

3	Total expenditure by vendor in a given year.	<pre>SELECT DISTINCT V.VendorID, FORMAT(SUM(Price), 'C', 'en-us') AS Total, YEAR(Date) AS Year FROM T_Vendor V, T_POLine POL, T_PurchaseOrder PO WHERE V.VendorID = POL.VendorID AND POL.POID = PO.POID AND YEAR(Date) = 2018 GROUP BY V.VendorID, YEAR(Date)</pre>	<table><tr><th></th><th>VendorID</th><th>Total</th><th>Year</th></tr><tr><td>1</td><td>1</td><td>\$5,400.00</td><td>2018</td></tr><tr><td>2</td><td>3</td><td>\$2,700.00</td><td>2018</td></tr><tr><td>3</td><td>8</td><td>\$1,100.00</td><td>2018</td></tr></table>		VendorID	Total	Year	1	1	\$5,400.00	2018	2	3	\$2,700.00	2018	3	8	\$1,100.00	2018																								
	VendorID	Total	Year																																								
1	1	\$5,400.00	2018																																								
2	3	\$2,700.00	2018																																								
3	8	\$1,100.00	2018																																								
4	The ten highest selling models	<pre>SELECT DISTINCT ProductName, SUM(Qty) AS UnitsSold FROM T_Product P, T_SalesOrderLine SOL WHERE P.ProductID = SOL.ProductID GROUP BY ProductName ORDER BY UnitsSold DESC</pre>	<table><tr><th></th><th>ProductName</th><th>UnitsSold</th></tr><tr><td>1</td><td>Sirius</td><td>90</td></tr><tr><td>2</td><td>Thunderbird</td><td>90</td></tr><tr><td>3</td><td>Thunder Bx</td><td>60</td></tr><tr><td>4</td><td>Boxer</td><td>40</td></tr><tr><td>5</td><td>Eagle</td><td>40</td></tr><tr><td>6</td><td>Kyon</td><td>35</td></tr></table>		ProductName	UnitsSold	1	Sirius	90	2	Thunderbird	90	3	Thunder Bx	60	4	Boxer	40	5	Eagle	40	6	Kyon	35																			
	ProductName	UnitsSold																																									
1	Sirius	90																																									
2	Thunderbird	90																																									
3	Thunder Bx	60																																									
4	Boxer	40																																									
5	Eagle	40																																									
6	Kyon	35																																									
5	The ten highest selling terrain ski types	<pre>SELECT DISTINCT Description AS TerrainType, SUM(Qty) AS UnitsSold FROM T_Product P, T_SalesOrderLine SOL, T_Terrain T WHERE P.ProductID = SOL.ProductID AND T.TerrainID = P.TerrainTypeID GROUP BY Description ORDER BY UnitsSold DESC</pre>	<table><tr><th></th><th>TerrainType</th><th>UnitsSold</th></tr><tr><td>1</td><td>Smooth</td><td>220</td></tr><tr><td>2</td><td>Halfpipe</td><td>60</td></tr><tr><td>3</td><td>Track</td><td>40</td></tr><tr><td>4</td><td>Cross-country</td><td>35</td></tr></table>		TerrainType	UnitsSold	1	Smooth	220	2	Halfpipe	60	3	Track	40	4	Cross-country	35																									
	TerrainType	UnitsSold																																									
1	Smooth	220																																									
2	Halfpipe	60																																									
3	Track	40																																									
4	Cross-country	35																																									
6	The ten highest selling model-sizes.	<pre>SELECT DISTINCT ProductShape, SUM(Qty) AS UnitsSold FROM T_Product P, T_SalesOrderLine SOL WHERE P.ProductID = SOL.ProductID GROUP BY ProductShape ORDER BY UnitsSold DESC</pre>	<table><tr><th></th><th>ProductShape</th><th>UnitsSold</th></tr><tr><td>1</td><td>Sport</td><td>165</td></tr><tr><td>2</td><td>Universal</td><td>150</td></tr><tr><td>3</td><td>Race</td><td>40</td></tr></table>		ProductShape	UnitsSold	1	Sport	165	2	Universal	150	3	Race	40																												
	ProductShape	UnitsSold																																									
1	Sport	165																																									
2	Universal	150																																									
3	Race	40																																									
7	The number of distinct products managed by each product manager	<pre>SELECT DISTINCT E.EmployeeID, FirstName, LastName, COUNT(TerrainID) AS NumberManaged FROM T_Terrain T, T_Employee E WHERE T.EmployeeID = E.EmployeeID GROUP BY E.EmployeeID, FirstName, LastName</pre>	<table><tr><th></th><th>EmployeeID</th><th>FirstName</th><th>LastName</th><th>NumberManaged</th></tr><tr><td>1</td><td>3</td><td>Canon</td><td>Frye</td><td>2</td></tr><tr><td>2</td><td>4</td><td>Sonic</td><td>Hedgehog</td><td>2</td></tr><tr><td>3</td><td>5</td><td>Amanda</td><td>Walker</td><td>1</td></tr><tr><td>4</td><td>6</td><td>Ricky</td><td>Mortimer</td><td>1</td></tr><tr><td>5</td><td>7</td><td>Terry</td><td>Crews</td><td>1</td></tr><tr><td>6</td><td>8</td><td>Martin</td><td>Gantx</td><td>2</td></tr><tr><td>7</td><td>9</td><td>Lord</td><td>Farquard</td><td>1</td></tr></table>		EmployeeID	FirstName	LastName	NumberManaged	1	3	Canon	Frye	2	2	4	Sonic	Hedgehog	2	3	5	Amanda	Walker	1	4	6	Ricky	Mortimer	1	5	7	Terry	Crews	1	6	8	Martin	Gantx	2	7	9	Lord	Farquard	1
	EmployeeID	FirstName	LastName	NumberManaged																																							
1	3	Canon	Frye	2																																							
2	4	Sonic	Hedgehog	2																																							
3	5	Amanda	Walker	1																																							
4	6	Ricky	Mortimer	1																																							
5	7	Terry	Crews	1																																							
6	8	Martin	Gantx	2																																							
7	9	Lord	Farquard	1																																							
8	Total shipping costs for a given month by shipping type (freight, two-day, overnight).	<pre>SELECT DISTINCT Description, SUM(Cost) AS Cost, MONTH(ShipDate) AS Month FROM T_ShipmentOut SO, T_ShipType ST WHERE SO.ShipTypeID = ST.ShipTypeID AND MONTH(ShipDate) = 8 GROUP BY Description, MONTH(ShipDate) ORDER BY Cost DESC</pre>	<table><tr><th></th><th>Description</th><th>Cost</th><th>Month</th></tr><tr><td>1</td><td>Express</td><td>100.00</td><td>8</td></tr><tr><td>2</td><td>Ground</td><td>40.00</td><td>8</td></tr></table>		Description	Cost	Month	1	Express	100.00	8	2	Ground	40.00	8																												
	Description	Cost	Month																																								
1	Express	100.00	8																																								
2	Ground	40.00	8																																								

9	Purchase order requests (product manager name, purchase req number, item, quantity) that have been rejected by purchasing within a given year.	<pre>SELECT PO.POID, POL.POLID, EmpIDVerify, FirstName, LastName, RM.Name, POL.Qty, IsAccepted, YEAR(Date) AS Year FROM T_Employee E, T_PurchaseOrder PO, T_POLine POL, T_RawMaterial RM WHERE E.EmployeeID = PO.EmpIDVerify AND PO.POID = POL.POID AND POL.RawMatID = RM.RawMatID AND IsAccepted = 'Denied' AND YEAR(Date) = 2018</pre>	<table><tr><th></th><th>POID</th><th>POLID</th><th>EmpIDVerify</th><th>FirstName</th><th>LastName</th><th>Name</th><th>Qty</th><th>IsAccepted</th><th>Year</th></tr><tr><td>1</td><td>8</td><td>4</td><td>5</td><td>Amanda</td><td>Walker</td><td>Pine</td><td>1645</td><td>Denied</td><td>2018</td></tr><tr><td>2</td><td>8</td><td>6</td><td>5</td><td>Amanda</td><td>Walker</td><td>Oak</td><td>150</td><td>Denied</td><td>2018</td></tr><tr><td>3</td><td>5</td><td>9</td><td>6</td><td>Ricky</td><td>Motimer</td><td>Blue Paint</td><td>900</td><td>Denied</td><td>2018</td></tr></table>		POID	POLID	EmpIDVerify	FirstName	LastName	Name	Qty	IsAccepted	Year	1	8	4	5	Amanda	Walker	Pine	1645	Denied	2018	2	8	6	5	Amanda	Walker	Oak	150	Denied	2018	3	5	9	6	Ricky	Motimer	Blue Paint	900	Denied	2018																																					
	POID	POLID	EmpIDVerify	FirstName	LastName	Name	Qty	IsAccepted	Year																																																																							
1	8	4	5	Amanda	Walker	Pine	1645	Denied	2018																																																																							
2	8	6	5	Amanda	Walker	Oak	150	Denied	2018																																																																							
3	5	9	6	Ricky	Motimer	Blue Paint	900	Denied	2018																																																																							
10	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).	<pre>SELECT SO.SOID, SOL.Qty, SOL.ProductID, P.ProductName, FORMAT(P.ProductPrice, 'C', 'en-us'), FORMAT(SOL.Qty * P.ProductPrice, 'C', 'en-us') AS TotalPrice FROM T_SalesOrder SO, T_SalesOrderLine SOL, T_Product P WHERE SO.SOID = SOL.SOID AND SOL.ProductID = P.ProductID ORDER BY SO.SOID ASC</pre>	<table><tr><th></th><th>SOID</th><th>Qty</th><th>ProductID</th><th>ProductName</th><th>(No column name)</th><th>TotalPrice</th></tr><tr><td>1</td><td>1</td><td>15</td><td>5</td><td>Thunderbird</td><td>\$634.99</td><td>\$9,524.85</td></tr><tr><td>2</td><td>1</td><td>20</td><td>7</td><td>Sinus</td><td>\$749.99</td><td>\$14,999.80</td></tr><tr><td>3</td><td>2</td><td>50</td><td>5</td><td>Thunderbird</td><td>\$634.99</td><td>\$31,749.50</td></tr><tr><td>4</td><td>2</td><td>30</td><td>9</td><td>Boxer</td><td>\$699.99</td><td>\$20,999.70</td></tr><tr><td>5</td><td>2</td><td>35</td><td>8</td><td>Kyon</td><td>\$649.99</td><td>\$22,749.65</td></tr><tr><td>6</td><td>3</td><td>60</td><td>6</td><td>Thunder Bx</td><td>\$749.99</td><td>\$44,999.40</td></tr><tr><td>7</td><td>5</td><td>25</td><td>5</td><td>Thunderbird</td><td>\$634.99</td><td>\$15,874.75</td></tr><tr><td>8</td><td>6</td><td>10</td><td>9</td><td>Boxer</td><td>\$699.99</td><td>\$6,999.90</td></tr><tr><td>9</td><td>7</td><td>40</td><td>4</td><td>Eagle</td><td>\$659.99</td><td>\$26,399.60</td></tr><tr><td>10</td><td>8</td><td>70</td><td>7</td><td>Sinus</td><td>\$749.99</td><td>\$52,499.30</td></tr></table>		SOID	Qty	ProductID	ProductName	(No column name)	TotalPrice	1	1	15	5	Thunderbird	\$634.99	\$9,524.85	2	1	20	7	Sinus	\$749.99	\$14,999.80	3	2	50	5	Thunderbird	\$634.99	\$31,749.50	4	2	30	9	Boxer	\$699.99	\$20,999.70	5	2	35	8	Kyon	\$649.99	\$22,749.65	6	3	60	6	Thunder Bx	\$749.99	\$44,999.40	7	5	25	5	Thunderbird	\$634.99	\$15,874.75	8	6	10	9	Boxer	\$699.99	\$6,999.90	9	7	40	4	Eagle	\$659.99	\$26,399.60	10	8	70	7	Sinus	\$749.99	\$52,499.30
	SOID	Qty	ProductID	ProductName	(No column name)	TotalPrice																																																																										
1	1	15	5	Thunderbird	\$634.99	\$9,524.85																																																																										
2	1	20	7	Sinus	\$749.99	\$14,999.80																																																																										
3	2	50	5	Thunderbird	\$634.99	\$31,749.50																																																																										
4	2	30	9	Boxer	\$699.99	\$20,999.70																																																																										
5	2	35	8	Kyon	\$649.99	\$22,749.65																																																																										
6	3	60	6	Thunder Bx	\$749.99	\$44,999.40																																																																										
7	5	25	5	Thunderbird	\$634.99	\$15,874.75																																																																										
8	6	10	9	Boxer	\$699.99	\$6,999.90																																																																										
9	7	40	4	Eagle	\$659.99	\$26,399.60																																																																										
10	8	70	7	Sinus	\$749.99	\$52,499.30																																																																										
11	All model-sizes (regardless of whether a model-size has been sold) and, for those that have been sold, how many sales of each has taken place	<pre>SELECT DISTINCT P.ProductName, LengthDesc, P.QuantityonHand, SUM(SOL.Qty) AS QtySold FROM T_Product P LEFT JOIN T_SalesOrderLine SOL on P.ProductID = SOL.ProductID GROUP BY ProductName, LengthDesc, QuantityonHand</pre>	<table><tr><th></th><th>ProductName</th><th>LengthDesc</th><th>QuantityonHand</th><th>QtySold</th></tr><tr><td>1</td><td>Boxer</td><td>195</td><td>55</td><td>40</td></tr><tr><td>2</td><td>Eagle</td><td>164</td><td>50</td><td>40</td></tr><tr><td>3</td><td>Kyon</td><td>177</td><td>45</td><td>35</td></tr><tr><td>4</td><td>Osprey</td><td>180</td><td>35</td><td>NULL</td></tr><tr><td>5</td><td>Osprey FX</td><td>187</td><td>25</td><td>NULL</td></tr><tr><td>6</td><td>Raven</td><td>190</td><td>40</td><td>NULL</td></tr><tr><td>7</td><td>Rotty</td><td>174</td><td>75</td><td>NULL</td></tr><tr><td>8</td><td>Sinus</td><td>182</td><td>35</td><td>90</td></tr><tr><td>9</td><td>Thunder Bx</td><td>172</td><td>20</td><td>60</td></tr><tr><td>10</td><td>Thunderbird</td><td>186</td><td>60</td><td>90</td></tr></table>		ProductName	LengthDesc	QuantityonHand	QtySold	1	Boxer	195	55	40	2	Eagle	164	50	40	3	Kyon	177	45	35	4	Osprey	180	35	NULL	5	Osprey FX	187	25	NULL	6	Raven	190	40	NULL	7	Rotty	174	75	NULL	8	Sinus	182	35	90	9	Thunder Bx	172	20	60	10	Thunderbird	186	60	90																						
	ProductName	LengthDesc	QuantityonHand	QtySold																																																																												
1	Boxer	195	55	40																																																																												
2	Eagle	164	50	40																																																																												
3	Kyon	177	45	35																																																																												
4	Osprey	180	35	NULL																																																																												
5	Osprey FX	187	25	NULL																																																																												
6	Raven	190	40	NULL																																																																												
7	Rotty	174	75	NULL																																																																												
8	Sinus	182	35	90																																																																												
9	Thunder Bx	172	20	60																																																																												
10	Thunderbird	186	60	90																																																																												
12	Defect rate (i.e., number of units rejected after manufacturing) for a given model.	<pre>SELECT ProductName, QuantityonHand * DefectRate AS NumberDefective FROM T_Product</pre>	<table><tr><th></th><th>ProductName</th><th>NumberDefective</th></tr><tr><td>1</td><td>Raven</td><td>2.00</td></tr><tr><td>2</td><td>Osprey</td><td>1.75</td></tr><tr><td>3</td><td>Osprey FX</td><td>1.25</td></tr><tr><td>4</td><td>Eagle</td><td>2.50</td></tr><tr><td>5</td><td>Thunderbird</td><td>3.00</td></tr><tr><td>6</td><td>Thunder Bx</td><td>1.00</td></tr><tr><td>7</td><td>Sinus</td><td>1.75</td></tr><tr><td>8</td><td>Kyon</td><td>2.25</td></tr><tr><td>9</td><td>Boxer</td><td>2.75</td></tr><tr><td>10</td><td>Rotty</td><td>3.75</td></tr></table>		ProductName	NumberDefective	1	Raven	2.00	2	Osprey	1.75	3	Osprey FX	1.25	4	Eagle	2.50	5	Thunderbird	3.00	6	Thunder Bx	1.00	7	Sinus	1.75	8	Kyon	2.25	9	Boxer	2.75	10	Rotty	3.75																																												
	ProductName	NumberDefective																																																																														
1	Raven	2.00																																																																														
2	Osprey	1.75																																																																														
3	Osprey FX	1.25																																																																														
4	Eagle	2.50																																																																														
5	Thunderbird	3.00																																																																														
6	Thunder Bx	1.00																																																																														
7	Sinus	1.75																																																																														
8	Kyon	2.25																																																																														
9	Boxer	2.75																																																																														
10	Rotty	3.75																																																																														

13	A display of all ski products separated into short (under 150 cm), medium (150-175 cm), and long (over 175 cm) lengths.	<p>FOR SHORT PRODUCTS:</p> <pre>SELECT DISTINCT ProductName AS ShortProducts, LengthDesc FROM T_Product WHERE LengthDesc <= 150 GROUP BY ProductName, LengthDesc, QuantityonHand</pre> <p>FOR MEDIUM PRODUCTS:</p> <pre>SELECT DISTINCT ProductName AS MediumProducts, LengthDesc FROM T_Product WHERE LengthDesc >= 150 AND LengthDesc <= 175 GROUP BY ProductName, LengthDesc, QuantityonHand</pre> <p>FOR LONG PRODUCTS:</p> <pre>SELECT DISTINCT ProductName AS LongProducts, LengthDesc FROM T_Product WHERE LengthDesc >= 175 GROUP BY ProductName, LengthDesc, QuantityonHand</pre>	<table><thead><tr><th></th><th>MediumProducts</th><th>LengthDesc</th></tr></thead><tbody><tr><td>1</td><td>Eagle</td><td>164</td></tr><tr><td>2</td><td>Thunder Bx</td><td>172</td></tr><tr><td>3</td><td>Rotty</td><td>174</td></tr></tbody></table> <table><thead><tr><th></th><th>LongProducts</th><th>LengthDesc</th></tr></thead><tbody><tr><td>1</td><td>Kyon</td><td>177</td></tr><tr><td>2</td><td>Osprey</td><td>180</td></tr><tr><td>3</td><td>Sirius</td><td>182</td></tr><tr><td>4</td><td>Thunderbird</td><td>186</td></tr><tr><td>5</td><td>Osprey FX</td><td>187</td></tr><tr><td>6</td><td>Raven</td><td>190</td></tr><tr><td>7</td><td>Boxer</td><td>195</td></tr></tbody></table>		MediumProducts	LengthDesc	1	Eagle	164	2	Thunder Bx	172	3	Rotty	174		LongProducts	LengthDesc	1	Kyon	177	2	Osprey	180	3	Sirius	182	4	Thunderbird	186	5	Osprey FX	187	6	Raven	190	7	Boxer	195						
	MediumProducts	LengthDesc																																											
1	Eagle	164																																											
2	Thunder Bx	172																																											
3	Rotty	174																																											
	LongProducts	LengthDesc																																											
1	Kyon	177																																											
2	Osprey	180																																											
3	Sirius	182																																											
4	Thunderbird	186																																											
5	Osprey FX	187																																											
6	Raven	190																																											
7	Boxer	195																																											
14	List of all customers that have not made a purchase within the last 3 months from the current date.	<pre>SELECT DISTINCT C.CustomerID, FirstName, LastName FROM T_Customer C LEFT JOIN T_SalesOrder SO ON C.CustomerID = SO.CustomerID WHERE MONTH(Date) <= MONTH(GETDATE()) - 3</pre>	<table><thead><tr><th></th><th>CustomerID</th><th>FirstName</th><th>LastName</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Fred</td><td>Jones</td></tr><tr><td>2</td><td>3</td><td>Matthew</td><td>Smith</td></tr><tr><td>3</td><td>4</td><td>Frank</td><td>Ocean</td></tr><tr><td>4</td><td>7</td><td>Kevin</td><td>Brown</td></tr><tr><td>5</td><td>9</td><td>Ruby</td><td>Jewel</td></tr></tbody></table>		CustomerID	FirstName	LastName	1	1	Fred	Jones	2	3	Matthew	Smith	3	4	Frank	Ocean	4	7	Kevin	Brown	5	9	Ruby	Jewel																		
	CustomerID	FirstName	LastName																																										
1	1	Fred	Jones																																										
2	3	Matthew	Smith																																										
3	4	Frank	Ocean																																										
4	7	Kevin	Brown																																										
5	9	Ruby	Jewel																																										
15	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.	<pre>SELECT C.CustomerID, FirstName, LastName, AVG(SUM(SOL.Qty * P.ProductPrice)) OVER () AS Average, AVG(SOL.Qty * P.ProductPrice) AS PersonalAvg FROM T_Customer C, T_SalesOrder SO, T_SalesOrderLine SOL, T_Product P WHERE C.CustomerID = SO.CustomerID AND SO.SOID = SOL.SOID AND P.ProductID = SOL.ProductID GROUP BY C.CustomerID, FirstName, LastName</pre>	<table><thead><tr><th></th><th>CustomerID</th><th>FirstName</th><th>LastName</th><th>Average</th><th>PersonalAvg</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>Fred</td><td>Jones</td><td>41132.7416</td><td>52499.30</td></tr><tr><td>2</td><td>3</td><td>Matthew</td><td>Smith</td><td>41132.7416</td><td>6999.90</td></tr><tr><td>3</td><td>4</td><td>Frank</td><td>Ocean</td><td>41132.7416</td><td>25166.2833</td></tr><tr><td>4</td><td>6</td><td>Sarah</td><td>Franz</td><td>41132.7416</td><td>12262.325</td></tr><tr><td>5</td><td>7</td><td>Kevin</td><td>Brown</td><td>41132.7416</td><td>35699.50</td></tr><tr><td>6</td><td>8</td><td>Mary</td><td>Bell</td><td>41132.7416</td><td>15874.75</td></tr></tbody></table>		CustomerID	FirstName	LastName	Average	PersonalAvg	1	1	Fred	Jones	41132.7416	52499.30	2	3	Matthew	Smith	41132.7416	6999.90	3	4	Frank	Ocean	41132.7416	25166.2833	4	6	Sarah	Franz	41132.7416	12262.325	5	7	Kevin	Brown	41132.7416	35699.50	6	8	Mary	Bell	41132.7416	15874.75
	CustomerID	FirstName	LastName	Average	PersonalAvg																																								
1	1	Fred	Jones	41132.7416	52499.30																																								
2	3	Matthew	Smith	41132.7416	6999.90																																								
3	4	Frank	Ocean	41132.7416	25166.2833																																								
4	6	Sarah	Franz	41132.7416	12262.325																																								
5	7	Kevin	Brown	41132.7416	35699.50																																								
6	8	Mary	Bell	41132.7416	15874.75																																								

Three Additional Queries

Here are some questions we came up with that we think PSC will find helpful in more general situations. While these SQL statements are more simple, they help provide a baseline for further research using the database. If something can be seen as abnormal or patterns are found in this data, it may help PSC in making more broad decisions.

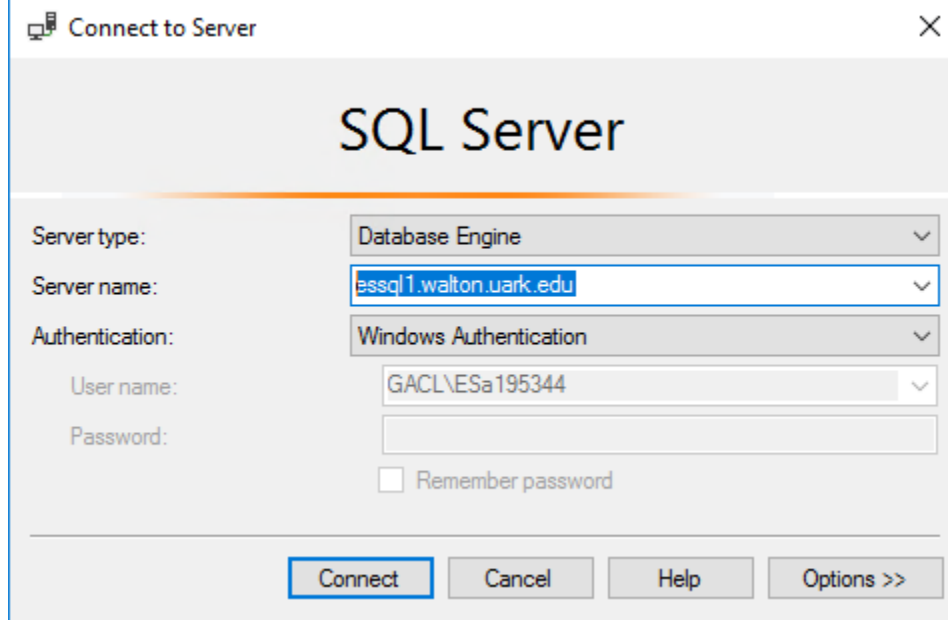
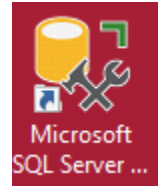
Query #	Question	Why is this important	SQL	Partial Output	Recap of Findings																																																																																																			
1	What is our current product inventory?	This is needed to check if more parts need to be ordered, or more products of a specific type made.	<pre>SELECT ProductID, ProductName, QuantityonHand FROM T_Product P</pre>	<table><tr><th></th><th>ProductID</th><th>ProductName</th><th>QuantityonHand</th></tr><tr><td>1</td><td>1</td><td>Raven</td><td>40</td></tr><tr><td>2</td><td>2</td><td>Osprey</td><td>35</td></tr><tr><td>3</td><td>3</td><td>Osprey FX</td><td>25</td></tr><tr><td>4</td><td>4</td><td>Eagle</td><td>50</td></tr><tr><td>5</td><td>5</td><td>Thunderbird</td><td>60</td></tr><tr><td>6</td><td>6</td><td>Thunder Bx</td><td>20</td></tr><tr><td>7</td><td>7</td><td>Sirius</td><td>35</td></tr><tr><td>8</td><td>8</td><td>Kyon</td><td>45</td></tr><tr><td>9</td><td>9</td><td>Boxer</td><td>55</td></tr><tr><td>10</td><td>10</td><td>Rotty</td><td>75</td></tr></table>		ProductID	ProductName	QuantityonHand	1	1	Raven	40	2	2	Osprey	35	3	3	Osprey FX	25	4	4	Eagle	50	5	5	Thunderbird	60	6	6	Thunder Bx	20	7	7	Sirius	35	8	8	Kyon	45	9	9	Boxer	55	10	10	Rotty	75	We can see we have a lot more of some types on hand than others. This may cause us to investigate how well each is selling and adjust production accordingly.																																																							
	ProductID	ProductName	QuantityonHand																																																																																																					
1	1	Raven	40																																																																																																					
2	2	Osprey	35																																																																																																					
3	3	Osprey FX	25																																																																																																					
4	4	Eagle	50																																																																																																					
5	5	Thunderbird	60																																																																																																					
6	6	Thunder Bx	20																																																																																																					
7	7	Sirius	35																																																																																																					
8	8	Kyon	45																																																																																																					
9	9	Boxer	55																																																																																																					
10	10	Rotty	75																																																																																																					
2	Provide all information in a purchase order.	This keeps track of the money PSC is spending	<pre>SELECT PO.*, POL.Qty, POL.Price, RM.Name AS MaterialName FROM T_PurchaseOrder PO, T_POLine POL, T_RawMaterial RM WHERE PO.POID = POL.POID AND POL.RawMatID = RM.RawMatID</pre>	<table><tr><th></th><th>POID</th><th>EmpIDverfy</th><th>EmpIDsubmt</th><th>Date</th><th>IsAccepted</th><th>Qty</th><th>Price</th><th>MaterialName</th></tr><tr><td>1</td><td>1</td><td>8</td><td>6</td><td>2018-10-30</td><td>Accepted</td><td>100</td><td>500.00</td><td>Pine</td></tr><tr><td>2</td><td>6</td><td>5</td><td>3</td><td>2018-07-06</td><td>Accepted</td><td>200</td><td>200.00</td><td>Red Paint</td></tr><tr><td>3</td><td>9</td><td>8</td><td>1</td><td>2018-12-20</td><td>Accepted</td><td>400</td><td>600.00</td><td>Sheet Metal</td></tr><tr><td>4</td><td>8</td><td>5</td><td>7</td><td>2018-06-15</td><td>Denied</td><td>1645</td><td>5000.00</td><td>Pine</td></tr><tr><td>5</td><td>6</td><td>5</td><td>3</td><td>2018-07-06</td><td>Accepted</td><td>200</td><td>600.00</td><td>Redwood</td></tr><tr><td>6</td><td>8</td><td>5</td><td>7</td><td>2018-06-15</td><td>Denied</td><td>150</td><td>300.00</td><td>Oak</td></tr><tr><td>7</td><td>2</td><td>3</td><td>1</td><td>2018-08-15</td><td>Accepted</td><td>50</td><td>500.00</td><td>Cedar</td></tr><tr><td>8</td><td>7</td><td>3</td><td>6</td><td>2018-02-09</td><td>Accepted</td><td>100</td><td>300.00</td><td>Cedar</td></tr><tr><td>9</td><td>5</td><td>6</td><td>3</td><td>2018-06-10</td><td>Denied</td><td>900</td><td>200.00</td><td>Blue Paint</td></tr><tr><td>10</td><td>9</td><td>8</td><td>1</td><td>2018-12-20</td><td>Accepted</td><td>1000</td><td>1000.00</td><td>Oak</td></tr></table>		POID	EmpIDverfy	EmpIDsubmt	Date	IsAccepted	Qty	Price	MaterialName	1	1	8	6	2018-10-30	Accepted	100	500.00	Pine	2	6	5	3	2018-07-06	Accepted	200	200.00	Red Paint	3	9	8	1	2018-12-20	Accepted	400	600.00	Sheet Metal	4	8	5	7	2018-06-15	Denied	1645	5000.00	Pine	5	6	5	3	2018-07-06	Accepted	200	600.00	Redwood	6	8	5	7	2018-06-15	Denied	150	300.00	Oak	7	2	3	1	2018-08-15	Accepted	50	500.00	Cedar	8	7	3	6	2018-02-09	Accepted	100	300.00	Cedar	9	5	6	3	2018-06-10	Denied	900	200.00	Blue Paint	10	9	8	1	2018-12-20	Accepted	1000	1000.00	Oak	Here we can see which orders are accepted and denied along with what was ordered, price, quantity, and other useful information.
	POID	EmpIDverfy	EmpIDsubmt	Date	IsAccepted	Qty	Price	MaterialName																																																																																																
1	1	8	6	2018-10-30	Accepted	100	500.00	Pine																																																																																																
2	6	5	3	2018-07-06	Accepted	200	200.00	Red Paint																																																																																																
3	9	8	1	2018-12-20	Accepted	400	600.00	Sheet Metal																																																																																																
4	8	5	7	2018-06-15	Denied	1645	5000.00	Pine																																																																																																
5	6	5	3	2018-07-06	Accepted	200	600.00	Redwood																																																																																																
6	8	5	7	2018-06-15	Denied	150	300.00	Oak																																																																																																
7	2	3	1	2018-08-15	Accepted	50	500.00	Cedar																																																																																																
8	7	3	6	2018-02-09	Accepted	100	300.00	Cedar																																																																																																
9	5	6	3	2018-06-10	Denied	900	200.00	Blue Paint																																																																																																
10	9	8	1	2018-12-20	Accepted	1000	1000.00	Oak																																																																																																

3	List our cutomers from highest to lowest in by money spent.	This shows your most and least reliable customers	<pre>SELECT DISTINCT C.CustomerID, FirstName, LastName, SUM(SOL.Qty * P.ProductPrice) AS TotalPurchases FROM T_Customer C, T_SalesOrder SO, T_SalesOrderLine SOL, T_Product P WHERE C.CustomerID = SO.CustomerID AND SO.SOID = SOL.SOID AND SOL.ProductID = P.ProductID GROUP BY C.CustomerID, FirstName, LastName ORDER BY TotalPurchases DESC</pre>	<table><tr><th></th><th>CustomerID</th><th>FirstName</th><th>LastName</th><th>TotalPurchases</th></tr><tr><td>1</td><td>4</td><td>Frank</td><td>Ocean</td><td>75498.85</td></tr><tr><td>2</td><td>7</td><td>Kevin</td><td>Brown</td><td>71399.00</td></tr><tr><td>3</td><td>1</td><td>Fred</td><td>Jones</td><td>52499.30</td></tr><tr><td>4</td><td>6</td><td>Sarah</td><td>Franz</td><td>24524.65</td></tr><tr><td>5</td><td>8</td><td>Mary</td><td>Bell</td><td>15874.75</td></tr><tr><td>6</td><td>3</td><td>Matthew</td><td>Smith</td><td>6999.90</td></tr></table>		CustomerID	FirstName	LastName	TotalPurchases	1	4	Frank	Ocean	75498.85	2	7	Kevin	Brown	71399.00	3	1	Fred	Jones	52499.30	4	6	Sarah	Franz	24524.65	5	8	Mary	Bell	15874.75	6	3	Matthew	Smith	6999.90	Frank Ocean and Kevin Brown are our two best customers, as those two alone spend more money that everyone else combined. This shows they are loyal and we might offer them discounts in the future, or we may market harder towards the names on the lower end of the list.
	CustomerID	FirstName	LastName	TotalPurchases																																				
1	4	Frank	Ocean	75498.85																																				
2	7	Kevin	Brown	71399.00																																				
3	1	Fred	Jones	52499.30																																				
4	6	Sarah	Franz	24524.65																																				
5	8	Mary	Bell	15874.75																																				
6	3	Matthew	Smith	6999.90																																				

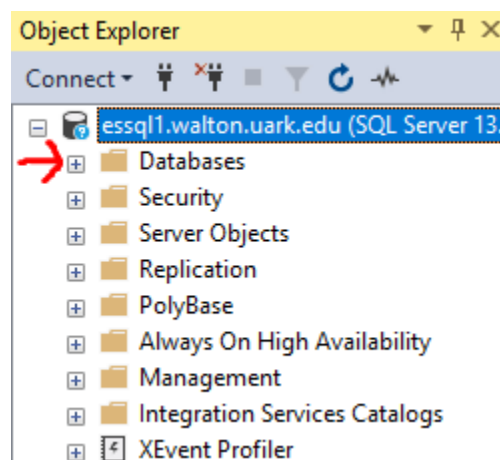
User Documentation

HOW TO ACCESS THE DATABASE

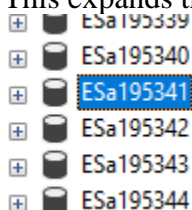
1. On the Desktop, click on the Microsoft SQL Server Management Studio Icon
2. Once opened, you will be asked to input a Server Type, Server Name, and Authentication. Make sure the fields are filled in as follows:



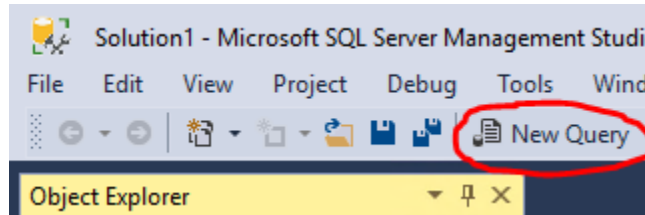
3. Click Connect
4. On the left side of the screen, there is the Object Explorer. Press the + sign to the left of the word Databases in the Object Explorer



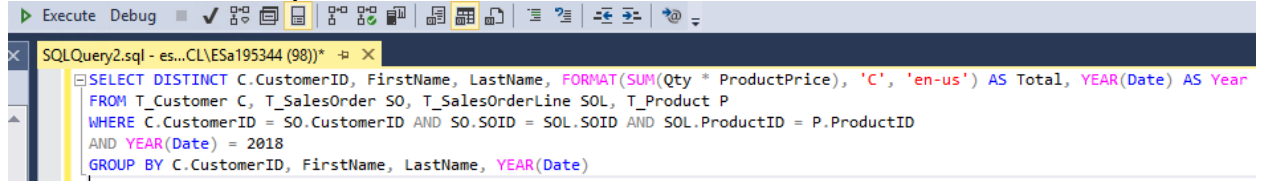
5. This expands the folder. Once Expanded, scroll until you find our database ESa195341



- Once highlighted in blue, click the New Query button just above the top right of the Object Explorer box



- You may now enter the SQL Statements we have provided and edit them to your satisfaction. For example:



- Once the Statement is typed, click the execute button shown in the top left of the previous picture. It will output the data below the statement as shown here:

	CustomerID	FirstName	LastName	Total	Year
1	1	Fred	Jones	\$52,499.30	2018
2	3	Matthew	Smith	\$6,999.90	2018
3	4	Frank	Ocean	\$75,498.85	2018
4	6	Sarah	Franz	\$24,524.65	2018
5	7	Kevin	Brown	\$71,399.00	2018
6	8	Mary	Bell	\$15,874.75	2018

What We Learned Throughout This Process

This process has taught Space Jam a great deal because it was our first project as a team and we discovered our strengths. We did not know what to expect through out this process but we enjoyed working as a team to accomplish this task. Each member learned new things about how to create a database and work as a team. This project also taught us how to manage our time and plan to maximize our efficiency.

During this project we learned about how to work together on a team project and pick up the slack of other members when we had to endure adversity by losing team members. The project management tool was beneficial because it showed us how efficient we were spending our time working on our project as well as how many hours each member allocated to the project. The project management tool also taught us how to properly manage our time because we could tell if we were ahead or behind schedule by looking at the schedule performance index and interpreting its value.

Member Name:	What you learned:
Canon Frye	<ul style="list-style-type: none"> • How to use a project management tool. • How to interpret the CPI and SPI for a project. • How to split up work and manage time efficiently. • How to create an ERD with production, revenue and expenditure cycles • How to normalize and denormalize an ERD. • How to create a data dictionary from an ERD.
Reid Harrington	<ul style="list-style-type: none"> • How to use SQL Server Management Studio. • How to manually create tables and add data to them. • How to manually edit already created tables through SQL statements. • How to implement a physical design. • How to coordinate as a team to make sure all tasks are completed in a timely manner.

Appendix

Team Contract

When Space Jam was started our team signed a contract to ensure and uphold that every team member would contribute an equal amount of work into our project and communicate efficiently as we proceeded with our project. We have been loyal to our contract and communicate via GroupMe constantly about what needs to be done, and we also divide work between us by choosing the parts of the project that are our strong suit. This has proven to be an effective style for us and we have enjoyed our time as team Space Jam.

Data Dictionary Model

<u>Table</u>	<u>Field Name</u>	<u>Data Type</u>	<u>Size</u>	<u>Null</u>	<u>Default</u>	<u>References</u>
T_Employee	EmployeeID*	Int (Auto-Increment)		Not Null		
	Name	Varchar	50			
	Position	Varchar	50			
T_ShipmentOut	ShipmentID*	Int (Auto-Increment)		Not Null		
	EmpID-Pick	Int				T_Employee
	EmpID-Pack	Int				T_Employee
	EmpID-Ship	Int				T_Employee
	ShipDate	Datestamp				
	CarrierID	Int				T_Carrier
	ShipStatus	Varchar				
	ShipTypeID	Int				T_ShipType
T_ShipType	ShipTypeID*	Int (Auto-Increment)		Not Null		
	Description	Text				
	Cost	money				
T_Carrier	CarrierID*	Int (Auto-Increment)		Not Null		
	Name	Varchar	50			
	Address	Varchar				
	Phone	Varchar				
T_SalesOrderLine	SOLID*	Int (Auto-Increment)		Not Null		
	DeliveryID	Int				T_DeliveryIn

<u>Table</u>	<u>Field Name</u>	<u>Data Type</u>	<u>Size</u>	<u>Null</u>	<u>Default</u>	<u>References</u>
	ProductID	Int				T_Product
	SOID	Int				T_SalesOrder
	Qty	Int				
	SOLTotal	Int				
	SOLStatus	Varchar				
	SaleOrReturn	Varchar				
T_SalesOrder	SOID*	Int (Auto-Increment)		Not Null		
	CustomerID	Int				T_Customer
	EmployeeID	Int				T_Employee
	PurchaseAddress	Varchar				
	Date	Datestamp				
	Status	Varchar				
	Total	money				
T_Customer	CustomerID*	Int (Auto-Increment)		Not Null		
	EmployeeID	Int				T_Employee
	RegionID	Int				T_Region
	Name	Varchar	50			
	Address	Varchar				
	Phone	Varchar				
	TermsID	Int				T_Terms
	IsOpenInvoice	Varchar				
	Size	Int				
	MeetDate	Datestamp				
	TimeKnown	Int				
T_Terms	TermsID*	Int (Auto-Increment)		Not Null		
	Length	Int				
	Discount	Decimal				
	DiscountPeriod	Int				

<u>Table</u>	<u>Field Name</u>	<u>Data Type</u>	<u>Size</u>	<u>Null</u>	<u>Default</u>	<u>References</u>
T_DeliveryIn	DeliveryID*	Int (Auto-Increment)		Not Null		
	EmpID-Receive	Int				T_Employee
	EmpID-Inspect	Int				T_Employee
	Date	Datestamp				
T_POrderLine	POLID*	Int (Auto-Increment)		Not Null		
	POID	Int				T_Porder
	RawMatID	Int				T_RawMaterial
	DeliveryID	Int				T_DeliveryIn
	Qty	Int				
	Price	money				
T_POrder	POID*	Int (Auto-Increment)		Not Null		
	EmpID-Verify	Int				T_Employee
	EmpID-Submit	Int				T_Employee
	Date	Datestamp				
T_Vendor	VendorID*	Int (Auto-Increment)		Not Null		
	Address	Varchar				
T_VendorRM	VRMID*	Int (Auto-Increment)		Not Null		
	VendorID	Int				T_Vendor
	RawMatID	Int				T_RawMaterial
	Cost	money				
T_RawMaterial	RawMatID*	Int (Auto-Increment)		Not Null		
	Name	Varchar	50			
	Qty	Int				
	Description	Text				
T_RMBOM	RMBOMID*	Int (Auto-Increment)		Not Null		
	RawMatID	Int				T_RawMaterial
	Qty	Int				

<u>Table</u>	<u>Field Name</u>	<u>Data Type</u>	<u>Size</u>	<u>Null</u>	<u>Default</u>	<u>References</u>
T_Product	ProductID*	Int (Auto-Increment)		Not Null		
	ProductShape	Varchar				
	ProductName	Varchar	50			
	ProductPrice	money				
	QuantityOnHand	Int				
	ProficiencyID	Int				T_Proficiency
	TerrainTypeID	Int				T_Terrain
	GenderID	Int				T_Gender
	LengthID	Int				T_Length

Project Management

Cost Performance Index (CPI) is a measure of how financially effective and efficient the project is compared to the budget.

Schedule Performance Index (SPI) is the measure of how close a project is to being completed compared to the schedule.

Milestone 1

CPI— 2.00

SPI— 1.00

Milestone 2

CPI—3.57

SPI—1.27

Milestone 3

CPI—2.50

SPI—1.00

Final Project

CPI—1.23

SPI—2.22