University of Oklahoma

Team Supreme

Team Supreme

Chase Bonar, Talia Dean, Armin Garcia, Ryan Henry, Charlie Robinson

MIS 3353 – Database Management

Swetha Siripurapu

March 10, 2020

Executive Summary

Assistance League of Norman (AL) is a non-profit organization that focuses on bettering the lives of students in the Norman community. One of their most notable projects is Operation School Bell (OSB). Through the mission of Operation School Bell, Assistance League clothes nearly 1,600 students in the surrounding community. Our student consulting team, Team Supreme, has been tasked with creating a database-based approach for OSB to improve their data handling.

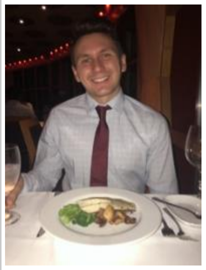
The purpose of this project proposal is to inform Assistance League management of the requirements, processes, and implementation of our database-based approach, as well as outline challenges our team faced throughout the project. First, we underwent the Conceptual Design phase in which we analyzed data from Assistance League documents and gained further insights from our client, Marty Giffin, during our client meeting on March 5, 2020. Based on the data and client requirements, we created an Entity-Relationship Diagram (ERD) using the Production, Revenue, and Expenditure business cycles. This visual representation served as the starting point for the next phase, Logical Design. During the Logical Design phase, our team arranged Assistance League's data into a series of logical relationships consisting of entities and attributes. Following completion of Logical Design, our team moved into the Physical Design process. This process involved integrating the logical design into the Relational Database Management System (RDBMS), which is SQL Server in this case. Once implemented, our team wrote SQL queries to provide accurate reports and data that were requested in the first phase. Finally, we have included detailed instructions for Assistance League staff to follow when accessing the database.

Our consulting team began this project on March 5, 2020 and the project was completed in full on April 26, 2020. Over the course of the project, our team adhered to the deadlines for each phase and adapted accordingly based on client feedback. A detailed report of task delegation and completion can be found in the final section of this report labeled as Project Management. Our team worked a total of 44.5 hours and the project cost totaled $1,113. While we faced challenges during this project, our team was able to overcome these and complete a database-based approach to fit the unique needs of Operation School Bell.

Contents

Get to Know the Team: Team Supreme

Name	Major	Year	Experience	Background	Photo
Chase Bonar	Supply Chain MS-IT	Senior	Supply Chain intern at Chesapeake Summer 2019 I-CCEW OFA intern Spring 2019	From Northlake, TX. I am a JCPLP member.	
Talia Dean	Accounting & MIS	Sophomore	Intern at ISN Software for the upcoming summer in 2020.	From Dallas, TX. Involved on campus in JCPLP, as well as an active member in Delta Gamma and MISSA.	
Armin Garcia	Industrial & Systems Engineering	Junior	Interning at Software company in Houston, TX for Summer 2020.	Originally born and raised in Mexico and moved to Houston, TX in the 4th grade.	

Name	Major	Year	Experience	Background	Photo
Ryan Henry	Civil Engineering	Junior	MIS-related projects with Alan Plummer Associates during the summer of 2019.	Originally born in Oman, moved to the United States at age 7. Lived in Tulsa until enrolling at OU where I started as a biochemistry major, then switched to civil engineering.	
Charlie Robinson	MIS	Senior	Interning for Software company	Born and raised in OKC, OK. Involved in OU's CAC as Vice Chair of Events for Speakers Bureau and as Social Media Coordinator for High School Leadership Conference.	

Conceptual Design

The Client Meeting

Operation School Bell (OSB) is a program created by Assistance League of Norman in order to assist financially disadvantaged children in Norman public schools by providing clothing for the school year. The project chairperson, Marty Giffin, would like a more effective database to be created to address the current needs of OSB. All members of Team Supreme including Chase Bonar, Talia Dean, Armin Garcia, Ryan Henry, and Charlie Robinson met with Marty Giffin to gather further information regarding the database requirements. The meeting was held in Adams Basement 7 Conference Room on March 5th, 2020 at 10:30am. Our team asked several questions in order to gain an understanding of OSB's objectives to be incorporated into our final product.

- Meeting Time: 3/5/2020 at 10:30am
- Location: Adams Basement 7 Conference Room
- Interviewers: Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry
- Interviewee: Marty Giffin

Q&A During the Meeting & Information We Learned

1. **How do you categorize the seasons of sales? Can children visit several times during a season?**

The seasons of sales are categorized by the length of the school year, in general. The goal of Assistance League is to prepare each child with enough clothing for the upcoming school year. It is usually the case that one visit per school year will provide enough durable clothing to last for the entire school year. However, children are not restricted from coming back to Assistance League if they grow out of their clothes. Assistance League should have the ability to report how many children have been serviced during a school year as well as whether a not a child visits multiple times throughout the year.

2. **Is the type of grooming kit based on child age or grade level? What is range for the level of grooming kit? Is the age range different depending on gender?**

Based on each child, Assistance League will individualize the child's grooming kit based on gender, age, necessary supplies, and supplies available on hand. There is not a one-size-fits-all grooming kit based on an age or grade level range. Regardless of gender however, grooming kits will include basic products. Additionally, children will not be restricted from any items due to their age or grade level, as every child develops at a different rate and will require different products.

3. **Would a volunteer service two children if they come in as a family?**

Each visit to Assistance League is a unique experience between one volunteer and one child. Although families may come in together and volunteers may help several children throughout the course of a day, children will be helped one at a time.

4. **How much growth are you expecting over each year?**

Each school receives an eligibility criterion outlining the number of students that can participate and that information is sent to Assistance League. The growth is dependent on the availability of resources as well as the eligibility determined by each school. Assistance League would also like to track the growth in students receiving help from their services.

5. **Should there be certain information kept about each school?**

Assistance League would like to keep track of what each school is bringing to the business in terms of number of children and the amount of inventory used. The counselors and teachers of these schools will recommend which children should benefit from the program. There is no need to track statistics of the school itself, but rather the number of children coming from each school.

Significant Assumptions

1. Assistance League has unlimited funds to support the completion of this project.
2. Volunteers may be responsible for receiving, inspecting, and sorting the deliveries. This relationship is modeled between the staff and delivery tables.
3. Grooming kits are created by each child and consist of the necessary components based on individual preference and item availability. There are multiple sizes of grooming kit bags to accommodate each child's different needs.
4. All products purchased by the vendor are clothing items. Other products such as books and toys have been donated to the organization.
5. Each school in the database has referred at least one student who has visited Assistance League.
6. The Raw Materials for the grooming kits are purchased from the Vendor.

What is an ERD? Why is it necessary?

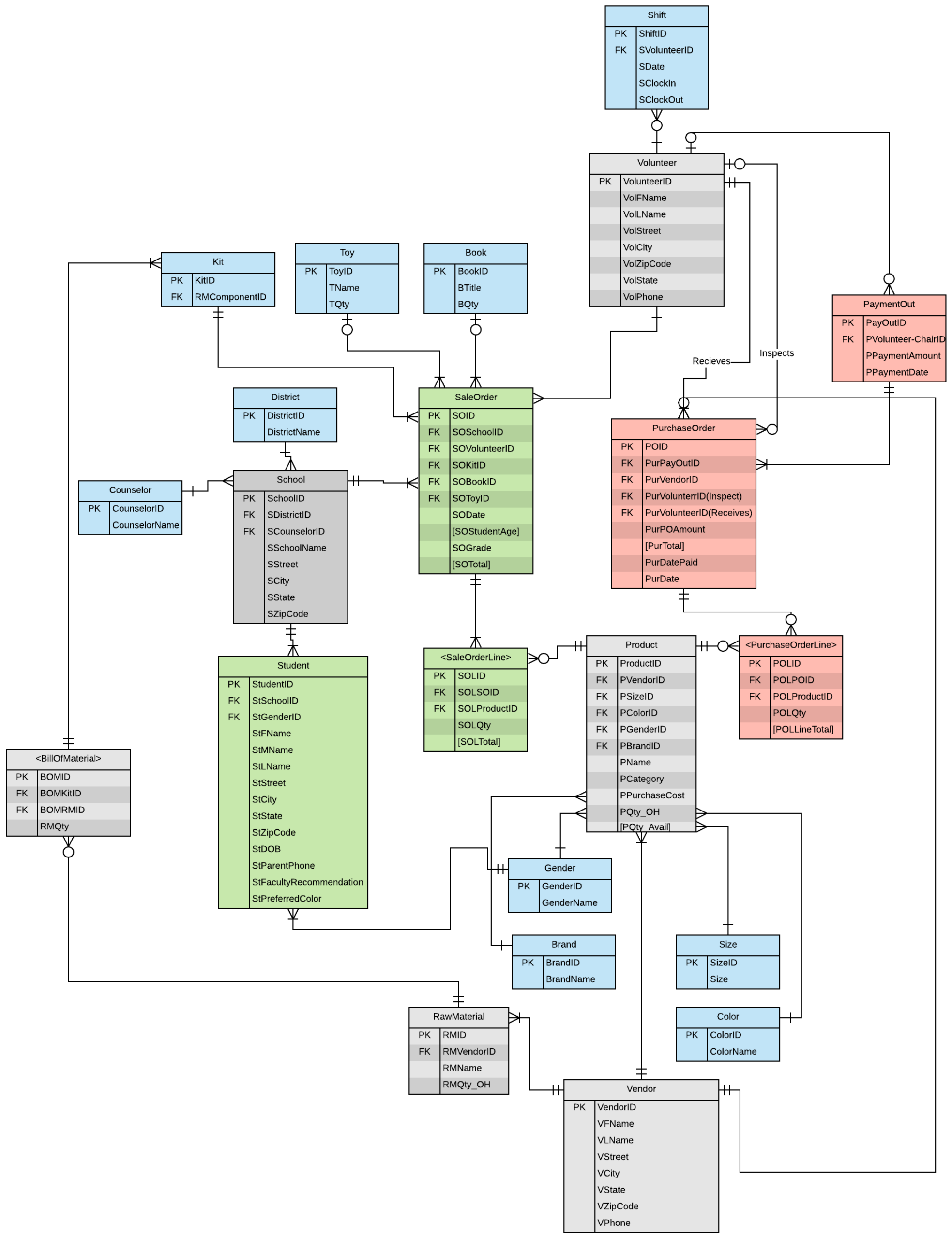
An Entity Relationship Diagram (ERD) is used to describe the relationship between multiple items and how they define the structure and organization of the database. If we were to build a house, the ERD would be the blueprint that the construction workers follow in order to build the house correctly. For this project, it's important to show the connection between the child, their school, the recommendation that brought them to Assistance League, and everything in between, so that we can organize and define the database in a concise manner.

Business Cycles Used

For this database, we used three business cycles: The Expenditure, Revenue, and Production cycles were used. The Expenditure Cycle is used to model the external exchange of information with suppliers or vendors. Assistance League purchases goods from a vendor in Las Vegas, receives the goods in Norman, and staff members inspect these deliveries. The Revenue Cycle is related to the information processing associated with providing goods and services to customers in exchange for cash. For Assistance League, we will be tracking the shopping experiences of children including the products they choose, and the prices associated with those products. By exchanging information with the Expenditure cycle, Assistance League can predict the amount of goods needed for each season, as well as any available inventory. Finally, the Production Cycle is concerned with the raw materials and components needed to produce an item. Assistance League produces kits for every child, so this is an essential cycle to have.

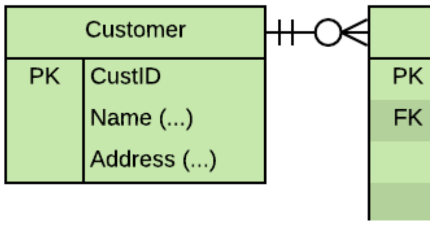
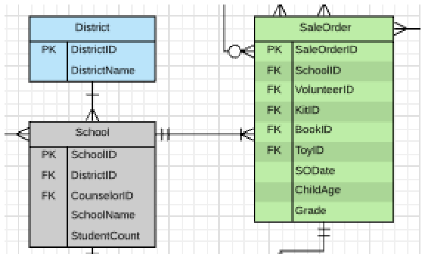
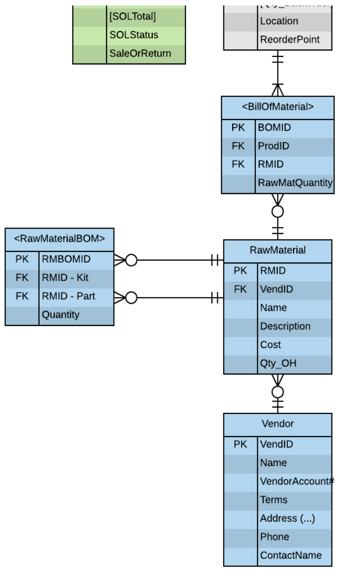
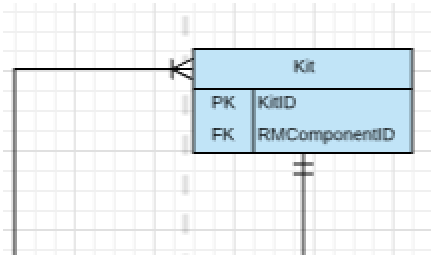
ERD Created

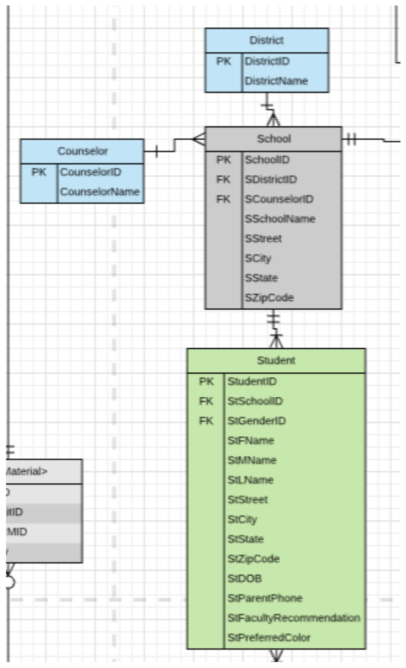
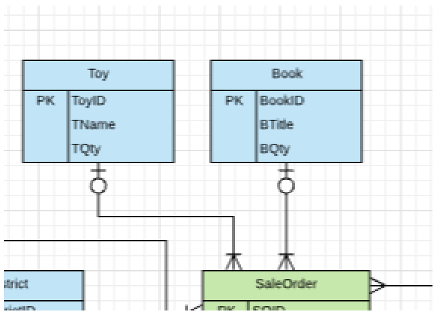
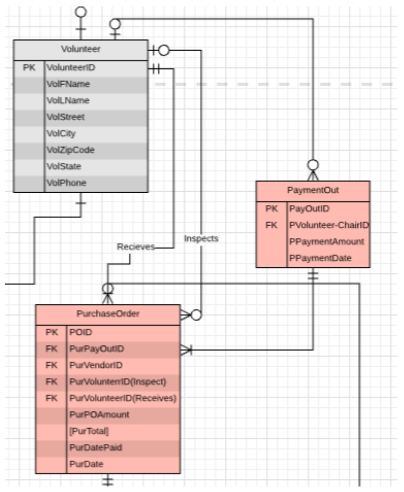
Below is the ERD our team has created to provide a visual starting point for the database design of Assistance League. When creating this design, our team considered the information given to us by Assistance League as well as the additional requirements defined by Marty Giffin. This ERD defines the information system requirements and relationships observed in the organization and will serve as a reference point for the Physical Design phase.

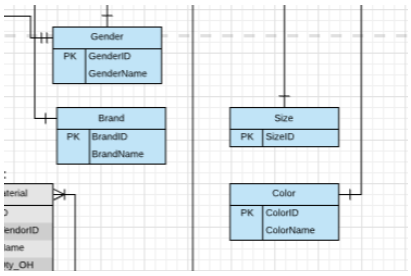
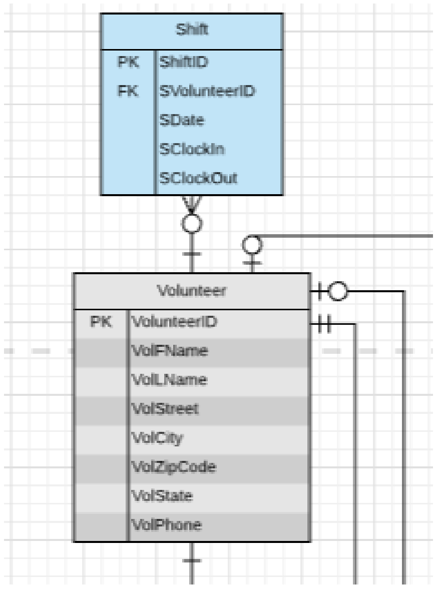


Changes made to Generic ERDs

For the creation of our ERD, we started with the generic integrated business cycle ERD. However, that generic ERD did not serve every single purpose that Assistance League requires, so our team made some changes to the generic diagrams. Detailed below are six of the changes and the impact that each change made on the database.

Change #	Original ERD	Updated ERD
#1: Changed the Customer entity to School. We added multiple attributes to fit the needs of Assistance League. The relationship between School and SaleOrder is one to many because each School in the database has had at least one student visit Assistance League.	 Customer PK CustID Name (...) Address (...) PK FK	 District PK DistrictID DistrictName School PK SchoolID FK DistrictID FK CounselorID SchoolName StudentCount SaleOrder PK SaleOrderID FK SchoolID FK VolunteerID FK KitID FK BookID FK ToyID SODate ChildAge Grade
#2: We added a Kit entity that connects to the RawMaterialsBOM entity as well as the Vendor entity. The Kit entity enables us to track the components of each child's kit which are considered to be the raw materials.	 [SOLTotal] SOLStatus SaleOrReturn Location ReorderPoint <BillOfMaterial> PK BOMID FK ProdID FK RMID RawMatQuantity <RawMaterialBOM> PK RMBOMID FK RMID - Kit FK RMID - Part Quantity RawMaterial PK RMID FK VendID Name Description Cost Qty_OH Vendor PK VendID Name VendorAccount# Terms Address (...) Phone ContactName	

#3: Added new reference table for school to track which children come from each school. Additionally, we have reference tables for District and Counselor in order to track this information.	This was not included in the original integrated business cycle.	 District PK DistrictID DistrictName Counselor PK CounselorID CounselorName School PK SchoolID FK SDistrictID FK SCounselorID SSchoolName SStreet SCity SState SZipCode Student PK StudentID FK StSchoolID FK StGenderID StFName StMName StLName StStreet StCity StState StZipCode StDOB StParentPhone StFacultyRecommendation StPreferredColor
#4: Added multiple reference tables to represent items that have been donated. These items are related to the SaleOrder directly, and are not part of the production cycle.	This was not included in the original integrated business cycle.	 Toy PK ToyID TName TQty Book PK BookID BTitle BQty SaleOrder
#5: Added new reference tables for deliveries and payments to track the reception, inspection, and payment of each delivery done by a volunteer. We denoted the volunteer's specific role within the attributes.	This was not included in the original integrated business cycle.	 Volunteer PK VolunteerID VolFName VolLName VolStreet VolCity VolZipCode VolState VolPhone PaymentOut PK PayOutID FK PVolunteer-ChairID PPaymentAmount PPaymentDate Recieves Inspects PurchaseOrder PK POID FK PurPayOutID FK PurVendorID FK PurVolunterrID(Inspect) FK PurVolunteerID(Receives) PurPOAmount [PurTotal] PurDatePaid PurDate

#6: Added many reference tables to the production cycle. These tables are essential to be able to process accurate reports.	This was not included in the original integrated business cycle.	 Gender PK GenderID GenderName Brand PK BrandID BrandName Size PK SizeID Color PK ColorID ColorName  Shift PK ShiftID FK SVolunteerID SDate SClockIn SClockOut Volunteer PK VolunteerID VolFName VolLName VolStreet VolCity VolZipCode VolState VolPhone

Logical Design

The logical design process involves using our conceptual understanding to arrange the data requirements of Assistance League into a series of logical relationships consisting of entities and attributes. This step of the design process will help us make sure that we are building a database with the capabilities to satisfy the needs of Assistance League. Additionally, careful logical design is important for the future physical design and implementation phase.

Normalization

Normalization is a process through which developers can help ensure that a data integrity, or in other words, the database is trustworthy. Data integrity is essential to be able to trust that the database reflects reality and is efficient. This is especially important to provide accurate reports for Operation School Bell and make certain that users experience efficient interactions with the database. Normalization required us to ensure atomicity of columns and eliminate data duplication issues.

The term “atomicity” refers to field values that cannot be broken down any further and are therefore “atomic” like atoms. For instance, the composite attribute of “Name(…)” could be broken into components such as First Name, Middle Name, and Last Name. Because the “Name(…)” will consist of two or more columns once implemented into the database, it should be normalized into its component parts. Data redundancy occurs when values for a given field are the same across multiple records in a given table. First, this would cause a data integrity issue. For example, if a person’s name was entered multiple times and spelled differently each time, certain reports would not return accurate data. Second, data redundancy reduces the efficiency of the database, resulting in a longer processing time for the user. If fields contain data that would be better represented as foreign keys instead, the processing time may increase.

One way to ensure that data is atomic and avoid data redundancy is the concept of normal form, which is used to indicate how far a table is from normalization based on various criteria. There are four normal forms: zero normal form (0NF), first normal form (1NF), second normal form (2NF), and third normal form (3NF). The higher the normal form, the more normalized a table is. To reach 3NF, a table’s columns must all be atomic (1NF), have no partial functional dependencies (2NF), and no transitive dependences. When a relation is in 3NF, it should be atomic and contain no data duplication issues.

Normalized Relations

Reference Tables

TSize(SizeID)
TColor(ColorID)
TGender(GenderID)
TKit(KitID, RMComponentID)
Foreign Key RMComponentID references TRawMaterial
Not Null
On Delete Restrict
TVendor(VendorID, VFName, VLName, VStreet, VCity, VZipCode, VState, VPhone)
Foreign Key VZipCode references TZip
Not Null
On Delete Restrict

Production Cycle

TRawMaterial(RMID, RMVendorID, RMName, RMQty_OH)
Foreign Key RMVendorID references TVendor
Not Null
On Delete Restrict
TBillOfMaterial(BOMID, BOMProductID, BOMRMID, RMQty)
Foreign Key BOMProductID references TProduct
Not Null
On Delete Restrict
Foreign Key BOMRMID references TRawMaterial
Not Null
On Delete Restrict
TProduct(ProductID, PVendorID, PSizeID, PColorID, PGenderID, PBrandID, PName, PCategory, PPurchaseCost, PQty_OH, PQty_Avail)
Foreign Key PVendorID references TVendor
Not Null
On Delete Restrict
Foreign Key PSizeID references TSize
Not Null
On Delete Restrict
Foreign Key PColorID references TColor
Not Null
On Delete Restrict
Foreign Key PGenderID references TGender
Not Null
On Delete Restrict
Foreign Key PBrandID references TBrand
Not Null
On Delete Restrict

Expenditure Cycle
TVolunteer(VolunteerID, VolFirstName,VolLastName,VolPhone, VolStreet, VolCity, VolZipCode, VolState)
TShift(ShiftID, SVolunteerID, SDate, SClockIn, SClockOut)
Foreign Key SVolunteerID references TVolunteer
Null Allowed
On Delete Restrict
TPaymentOut(PayoutID, PVolunteer-ChairID, PPaymentOut, PPaymentDate)
Foreign Key PayVolunteerChairID references TVolunteer
Null Allowed
On Delete Restrict
TPurchaseOrder(POID, PurPayOutID, PurVendorID, PurVolunteerID – Inspect, PurVolunteerID – Receives, PurDescription, PurPOAmount, PurTotal, PurDatePaid, PurDate)
Foreign Key PurPayoutID references TPaymentOut
Not Null
On Delete Restrict
Foreign Key PurVendorID references TVendor
Not Null
On Delete Restrict
Foreign Key PurVolunteerID-Inspect references TVolunteer
Not Null
On Delete Restrict
Foreign Key PurVolunteerID-Receivesreferences TVolunteer
Not Null
On Delete Restrict
TPurchaseOrderLine(POLID, POLPOID, POLProductID, POLQty, POLLineTotal)
Foreign Key POLPOID references TPurchaseOrder
Not Null
On Delete Restrict
Foreign Key POLProductID references TProduct
Not Null
On Delete Restrict

Revenue Cycle
TDistrict(DistrictID, DistrictName)
TCounselor(CounselorID, CounselorFName, CounselorLName)
TSaleOrder(SOID, SOSchoolID, SOVolunteerID, SOKitID, , SOBookID, SOToyID, SODate, SOStudentAge, SOGrade, SOTotal)
Foreign Key SOSchoolID references TSchool
Not Null
On Delete Restrict

Foreign Key SOVolunteerID references TVolunteer
Not Null
On Delete Restrict
Foreign Key SOKitID references TKit
Not Null
On Delete Restrict
Foreign Key SOBookID references TBook
Null Allowed
On Delete Restrict
Foreign Key SOToyID references TToy
Null Allowed
On Delete Restrict
TSchool(SchoolID, SDistrictID, SCounselorID, SSchoolName, SStreet, SCity, SState, SZipCode)
Foreign Key SDistrictID references TDistrictID
Not Null
On Delete Restrict
Foreign Key SCounselorID references TCounselor
Not Null
On Delete Restrict
TStudent(StudentID, SStchoolID, StGenderID, StFName, StMName, StLName, StStreet, StCity, StState, StZipCode, StDOB, StParentPhone, StFacultyRecommendation, StPrefferedColor)
Foreign Key StSchoolID references TSchool
Not Null
On Delete Restrict
Foreign Key StGenderID references TGender
Not Null
On Delete Restrict
Foreign Key StZipCode references TZip
Not Null
On Delete Restrict
TSalesOrderLine(SOLID, SOLSOID, SOLProductID, Qty, SOLTotal)
Foreign Key SOLSODID references TSaleOrder
Not Null
On Delete Restrict
Foreign Key SOLProductID references TProduct
Not Null
On Delete Restrict

Differences between ERD and Normalized Relations

As defined in an earlier section, an Entity Relationship Diagram (ERD) is used to describe the relationship between multiple items and how they define the structure and organization of the database. Normalization, on the other hand, is a process through which developers can help ensure that a data integrity, or in other words, the database is trustworthy. An ERD may contain columns within tables that are not yet “atomic”. Consider the previous example, of a column named “Name(…)”. In an ERD, this column would be represented as an attribute on a table and would be shown simply as “Name(…)”. However, in a Normalized Relation, this attribute would be broken into its component parts, such as “First Name”, “Middle Name”, and “Last Name”. Refer to the normalized relations above on the TVolunteer and TStudent tables to see how columns can be broken down further like this example.

Referential Integrity

When converting the ERD into Normalized Relations, there are three key rules to keep in mind. These are the three types of integrity constraints – Entity Integrity, Referential Integrity, and Entity Integrity. Entity Integrity requires that every table in the diagram has a primary key and this key must exist for all records, cannot be a null value, and cannot change over time. Referential Integrity refers to the association among entities. To satisfy this constraint, each relationship between entities must be represented with a foreign key that matches a valid primary key. The foreign key could be a null value if the mandatory side is an optional one. The final constraint, Domain Integrity, specifies that all values in a column must be in the same “domain”. In other words, all values must be of the specified data type for the column. This integrity constraint corresponds to data “atomicity” in that it forces us to split any composite attribute into its “atomic” components. Together, these three constraints ensure that the database is designed in a way that works efficiently and accurately for users.

Physical Design and Implementation

Once we completed the conceptual and logical design phases, we had to prepare to implement our database. The conceptual and logical design phases were what we call “platform-agnostic”, meaning that they are the same regardless of the Database Management System platform being used. The physical design phase on the other hand, is “platform-specific” and ours was implemented in the Microsoft SQL Server. Our goal of physical design was to identify the best way to integrate the logical design into the relational database management system, SQL Server in this case. Once we implemented our design, we were able to run SQL queries to provide accurate reports and information requested by the client.

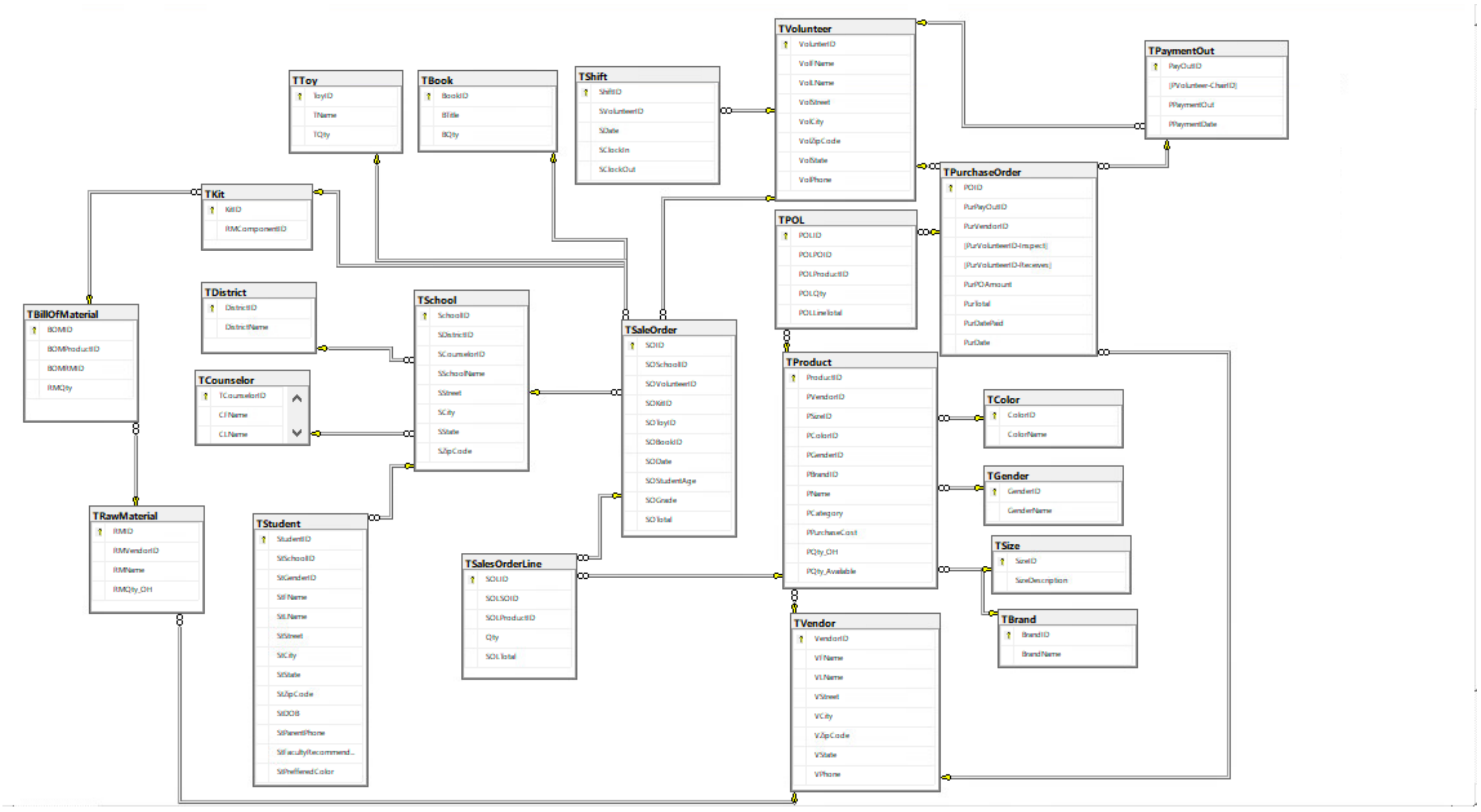
Data Dictionary

A data dictionary contains detailed information an insight into about the contents of a database. Field type, key type, data, and size are just a few the details used to describe the content. This dictionary can be used as a communication tool for the parties involved to outline and identify the details. This helps the database team design the database in a way that meets all the requirements of the customer.

Denormalization

Denormalization refers to the method of reducing normalization that was previously applied to the design. This is done by allowing certain data redundancy and duplication to occur within the database. The concept is that by decreasing or eliminating tables from the ERD, you can make the database process quicker and more efficiently. However, this process is counterintuitive to the normalization process. It is key to avoid denormalization of everything into one table because this causes the database to lose data integrity. This process was only used once to collapse the address tables into a larger table.

Implemented Physical Design



Challenges Faced/Addressed During Implementation

The group faced a few challenges when implementing the physical design into the relational database program:

1. Defining the data types throughout the design: with this being a larger database, it was incredibly important that the attributes were characterized as their proper data type. Our group was able to overcome this by spending time on our data dictionary. This gave us clarity and control over how the attributes were to be implemented.
2. Generating accurate data: our group struggled to generate data for every table. Specifically, we had to generate data in a way that would translate from one table entity to the next. The group was able to overcome this by focusing on the relations between each entity because this allowed us to generate data for tables that were dependent upon other table information.

Strengths and Weaknesses Encountered During Implementation

This was the first time for most of the group to work within RDBMS, so there was a large learning curve when trying to implement our design into the database.

Strengths:

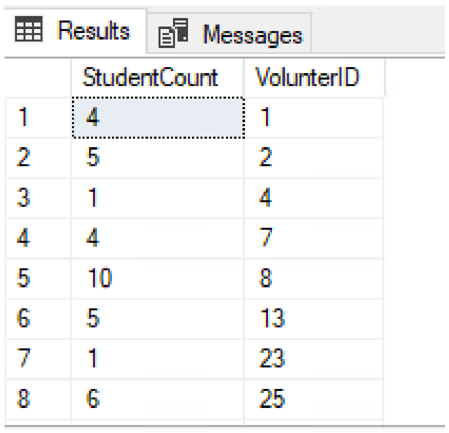
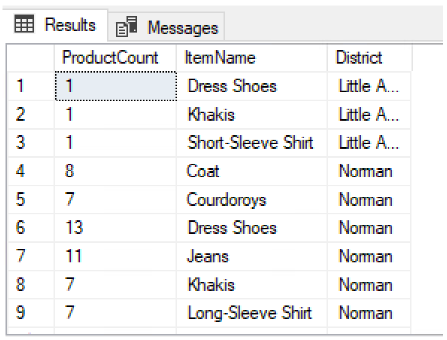
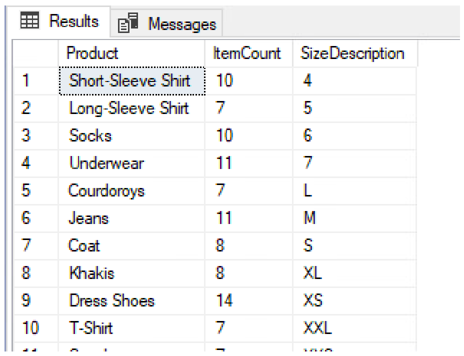
- Uploading data and creating accurate tables: The group was efficient in creating all the necessary tables needed for the database to function.

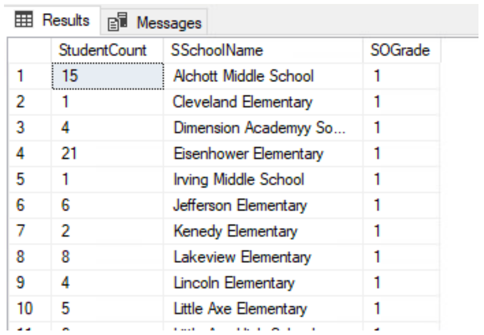
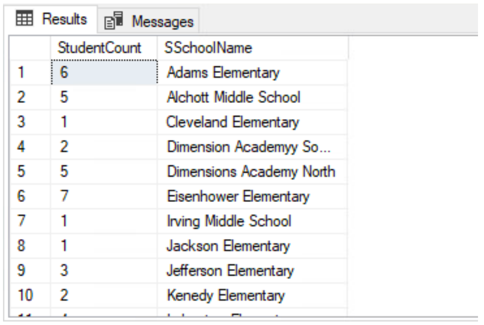
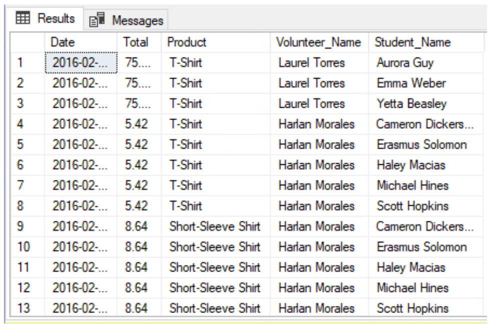
Weaknesses:

- Creating the relations within the database was difficult because it was tedious and required all our attributes to have the right data types. It was difficult to navigate the application so that we could properly assign the right size and data type to each attribute.
- SQL was a difficult process because most of the group was unfamiliar with how the query wizard worked within the application. The team has been used to typing out every query in the Select statement format. The application accomplished this in a manner that was unfamiliar.

Specific SQL Statements Requested

Our client had several requests for specific reports and data to be provided within the database. We have listed these requests below, along with the SQL queries that will be used to return the client's requested data.

Query #	Question	SQL	Partial Output
1	How many children were helped by each volunteer during a season?	SELECT DISTINCT COUNT(StudentID) as StudentCount FROM TStudent St join TSchool Sc on St.StSchoolID = Sc.SchoolID join TSaleOrder SO on Sc.SchoolID = SO.SOSchoolID join TVolunteer V on V.VolunteerID = SO.SOVolunteerID WHERE SODate Between (DateStart) AND (DateEnd) AND V.VolunteerID = ‘1’;	 Results Messages StudentCount VolunterID 1 4 1 2 5 2 3 1 4 4 4 7 5 10 8 6 5 13 7 1 23 8 6 25
2	How many of each item were distributed to the children from different school districts?	SELECT DISTINCT (P.PName) as ItemName, D.DistrictName FROM TProduct P join TSaleOrderLine SOL on P.ProductID = SOL.SOLProductID join TSaleOrder SO on SOL.SOLSOID = SO.SOID join TSchool Sc on SO.SOSchoolID = Sc.SchoolID join TDistrict D on Sc.DistrictID = D.DistrictID;	 Results Messages ProductCount ItemName District 1 1 Dress Shoes Little A... 2 1 Khakis Little A... 3 1 Short-Sleeve Shirt Little A... 4 8 Coat Norman 5 7 Courdoroys Norman 6 13 Dress Shoes Norman 7 11 Jeans Norman 8 7 Khakis Norman 9 7 Long-Sleeve Shirt Norman
3	How many items of each size were purchased by children? (this will help with ordering for the next season)	SELECT P.PName, COUNT(P.PName) as ItemCount, S.Size FROM TSize S join TProduct P on S.SizeID = P.ProductID join TSaleOrderLine SOL on P.ProductID = SOL.SOLProductID WHERE SOL.SOLID IS NOT NULL GROUP BY S.Size;	 Results Messages Product ItemCount SizeDescription 1 Short-Sleeve Shirt 10 4 2 Long-Sleeve Shirt 7 5 3 Socks 10 6 4 Underwear 11 7 5 Courdoroys 7 L 6 Jeans 11 M 7 Coat 8 S 8 Khakis 8 XL 9 Dress Shoes 14 XS 10 T-Shirt 7 XXL

4	How many children from each school were helped and what were their class grade? We want this information to be in one report.	SELECT DISTINCT COUNT(St.StudentID) as StudentCount, Sc.SSchoolName, SO.SOGrade FROM TSaleOrder SO join TSchool Sc on SO.SOSchoolID = Sc.SchoolID join TStudent St on Sc.SchoolID = St.StSchoolID GROUP BY Sc.SSchoolName, SO.SOGrade	
5	How many children from each school were helped and how many should we still expect to come?	SELECT DISTINCT COUNT(St.StudentID) as StudentCount, Sc.SSchoolName FROM TStudent St join TSchool Sc on Sc.SchoolID = St.StSchoolID GROUP BY Sc.SSchoolName	
6	An itemized invoice for each purchase.	SELECT DISTINCT SO.SODate as Date, SOL.SOLTotal as Total, P.PName as Product, CONCAT(Vol.VFName, ‘ ‘, Vol.VLName) as Volunteer_Name, CONCAT(St.StFName, ‘ ‘, St.StLName) as Student_Name FROM TProduct P join TSalesOrderLine SOL on P.ProductID = SOL.SOLProductID join TSaleOrder SO on SO.SOID = SOL.SOLSOID join TVolunteer V on V.VolunterID = SO.SOVolunteerID join TSchool Sc on Sc.SchoolID = SO.SOSchoolID join TStudent St on St.StSchoolID = Sc.SchoolID	

Results Messages

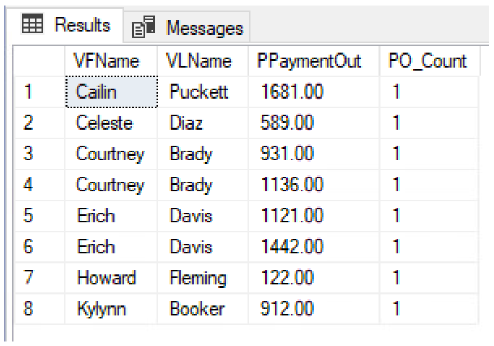
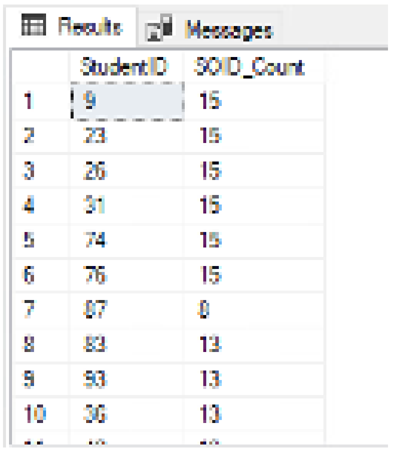
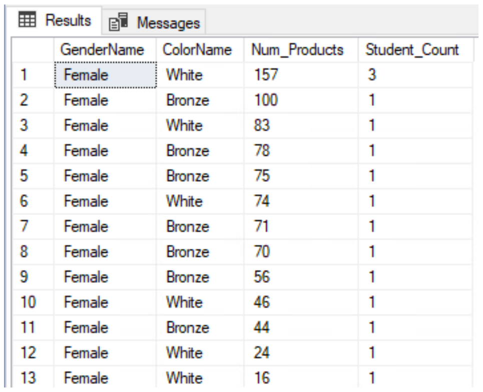
	StudentCount	SSchoolName	SOGrade
1	15	Alchott Middle School	1
2	1	Cleveland Elementary	1
3	4	Dimension Academyy So...	1
4	21	Eisenhower Elementary	1
5	1	Irving Middle School	1
6	6	Jefferson Elementary	1
7	2	Kenedy Elementary	1
8	8	Lakeview Elementary	1
9	4	Lincoln Elementary	1
10	5	Little Axe Elementary	1

Results Messages

	StudentCount	SSchoolName
1	6	Adams Elementary
2	5	Alchott Middle School
3	1	Cleveland Elementary
4	2	Dimension Academyy So...
5	5	Dimensions Academy North
6	7	Eisenhower Elementary
7	1	Irving Middle School
8	1	Jackson Elementary
9	3	Jefferson Elementary
10	2	Kenedy Elementary

Results Messages

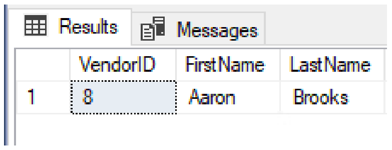
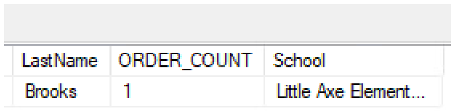
	Date	Total	Product	Volunteer_Name	Student_Name
1	2016-02-...	75....	T-Shirt	Laurel Torres	Aurora Guy
2	2016-02-...	75....	T-Shirt	Laurel Torres	Emma Weber
3	2016-02-...	75....	T-Shirt	Laurel Torres	Yetta Beasley
4	2016-02-...	5.42	T-Shirt	Harlan Morales	Cameron Dickers...
5	2016-02-...	5.42	T-Shirt	Harlan Morales	Erasmus Solomon
6	2016-02-...	5.42	T-Shirt	Harlan Morales	Haley Macias
7	2016-02-...	5.42	T-Shirt	Harlan Morales	Michael Hines
8	2016-02-...	5.42	T-Shirt	Harlan Morales	Scott Hopkins
9	2016-02-...	8.64	Short-Sleeve Shirt	Harlan Morales	Cameron Dickers...
10	2016-02-...	8.64	Short-Sleeve Shirt	Harlan Morales	Erasmus Solomon
11	2016-02-...	8.64	Short-Sleeve Shirt	Harlan Morales	Haley Macias
12	2016-02-...	8.64	Short-Sleeve Shirt	Harlan Morales	Michael Hines
13	2016-02-...	8.64	Short-Sleeve Shirt	Harlan Morales	Scott Hopkins

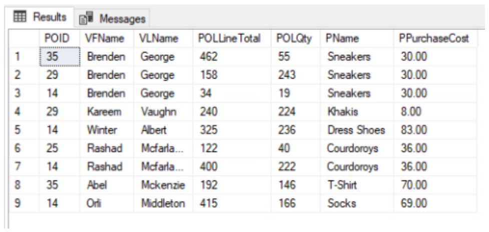
7	Who were our suppliers in a particular year? Provide the total paid for each supplier and the number of POs.	SELECT V.VLName, V.VFName, P.PPaymentAmount, COUNT(PO.POID) as PO_Count FROM TPaymentOut P join TPurchaseOrder PO on P.PayOutID = PO.PurPayOutID join TVendor V on PO.PurVendorID = V.VendorID WHERE YEAR(PO.PurDate) = '2019' GROUP BY V.VLName, V.VFName, P.PPaymentAmount	
8	How many children were served for more than one season?	SELECT COUNT(StudentID) as StudentCount FROM TSaleOrder SO join TSchool Sc on SO.SOSchoolID = Sc.SchoolID join TStudent St on Sc.SchoolID = St.StSchoolID GROUP BY St.StudentID HAVING COUNT(SO.SOID) > 1	
9	What is the preferred color of girls and boys? How many boys and girls did we serve?	SELECT DISTINCT G.GenderName, C.ColorName, SUM(SOL.Qty) as Num_products, COUNT(St.StudentID) as Student_Count FROM TGender G join TProduct P on G.GenderID = P.PGenderID join TColor C on C.ColorID = P.ColorID join TSalesOrderLine SOL on P.ProductID = SOL.SOLProductID join TSaleOrder SO on SO.SOID = SOL.SOLSOID join TSchool Sc on SO.SOSchoolID = St.StSchoolID WHERE G.GenderID = 1 OR G.GenderID = 2 GROUP BY C.ColorName, G.GenderName, St.StudentID ORDER BY G.GenderName, Num_Products DESC	

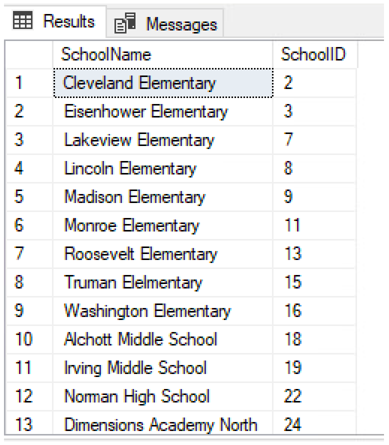
```
Results   Messages
   VFName    VLName   PPaymentOut  PO_Count
1  Cailin    Puckett  1681.00      1
2  Celeste   Diaz     589.00       1
3  Courtney  Brady    931.00       1
4  Courtney  Brady    1136.00      1
5  Erich     Davis    1121.00      1
6  Erich     Davis    1442.00      1
7  Howard    Fleming  122.00       1
8  Kylynn    Booker   912.00       1
```

```
Results   Messages
    StudentID  SOID_Count
1   9          15
2   23         15
3   26         15
4   31         15
5   74         15
6   76         15
7   87         0
8   83         13
9   93         13
10  36         13
```

```
Results   Messages
    GenderName  ColorName  Num_Products  Student_Count
1   Female      White      157           3
2   Female      Bronze     100           1
3   Female      White      83            1
4   Female      Bronze     78            1
5   Female      Bronze     75            1
6   Female      White      74            1
7   Female      Bronze     71            1
8   Female      Bronze     70            1
9   Female      Bronze     56            1
10  Female      White      46            1
11  Female      Bronze     44            1
12  Female      White      24            1
13  Female      White      16            1
```

10	What is the most purchased vendor by Little Axe Elementary Students?	SELECT DISTINCT V.VendorID, V.VFName as FirstName, V.VLName as LastName, COUNT(PO.POID) as ORDER_COUNT, Sc.SSchoolName as School
		FROM TVendor V join TPurchaseOrder PO on V.VendorID = PO.PurVendorID join TVolunteer Vt on Vt.VolunteerID = PO.[PurVolunteerID-Inspect] join TSaleOrder SO on SO.SOVolunteerID = Vt.VolunteerID join TSalesOrderLine SOL on SOL.SOLSOID = SO.SOID join TSchool SC on Sc.SchoolID = SO.SOSchoolID
		WHERE Sc.SSchoolName = ‘Little Axe Elementary’
		GROUP BY V.VendorID, V.VFName, V.VLName, S.SSchoolName
		ORDER BY ORDER_COUNT DESC
			 

11	For each PO that was placed in July of this year, show each line, including the PO number, name of vendor, name of product, item price, quantity, and line total.	SELECT PO.POID, V.VFName, V.VLName, POL.POLLineTotal, POL.POLQty, P.PName, P.PPurchaseCost
		FROM TPurchaseOrder PO join TPurchaseOrderLine POL on PO.POID = POL.POLPOID join TProduct P on P.ProductID = POL.POLProductID join TVendor V on V.VendorID = P.PVendorID
		WHERE MONTH(P.PurDate) = ‘July’
			

12	List all schools that did not send a child in a season?	SELECT DISTINCT Sc.SSchoolName as SchoolName, Sc.SchoolID FROM TSchool Sc WHERE Sc.SchoolID NOT IN (SELECT Sc.SchoolID FROM TStudent join TSchool Sc on St.StSchoolID = Sc.SchoolID join TSaleOrder SO on Sc.SchoolID = SO.SOSchoolID join TSaleOrderLine SOL on SO.SOID = SOL.SOLSOID WHERE SO.SODate BETWEEN ‘01-01-2018' AND ‘01-01-2019')	

Results Messages

	VendorID	FirstName	LastName
1	8	Aaron	Brooks

LastName	ORDER_COUNT	School
Brooks	1	Little Axe Element...

Results Messages

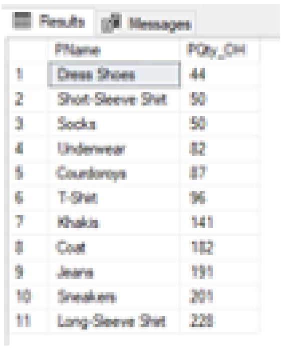
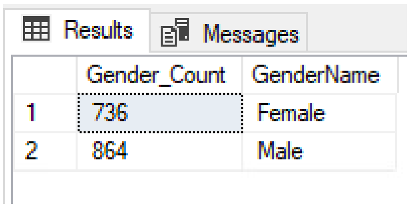
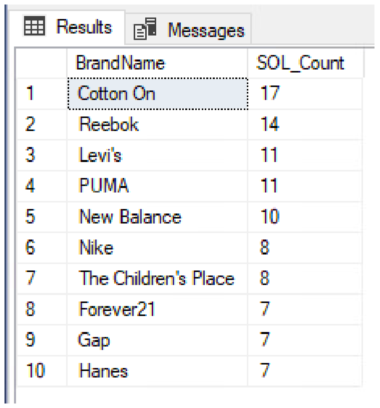
	POID	VFName	VLName	POLLineTotal	POLQty	PName	PPurchaseCost
1	35	Brenden	George	462	55	Sneakers	30.00
2	29	Brenden	George	158	243	Sneakers	30.00
3	14	Brenden	George	34	19	Sneakers	30.00
4	29	Kareem	Vaughn	240	224	Khakis	8.00
5	14	Winter	Albert	325	236	Dress Shoes	83.00
6	25	Rashad	Mcfarla...	122	40	Courdoroys	36.00
7	14	Rashad	Mcfarla...	400	222	Courdoroys	36.00
8	35	Abel	Mckenzie	192	146	T-Shirt	70.00
9	14	Orli	Middleton	415	166	Socks	69.00

Results Messages

	SchoolName	SchoolID
1	Cleveland Elementary	2
2	Eisenhower Elementary	3
3	Lakeview Elementary	7
4	Lincoln Elementary	8
5	Madison Elementary	9
6	Monroe Elementary	11
7	Roosevelt Elementary	13
8	Truman Elelmentary	15
9	Washington Elementary	16
10	Alchott Middle School	18
11	Irving Middle School	19
12	Norman High School	22
13	Dimensions Academy North	24

Three Additional Queries

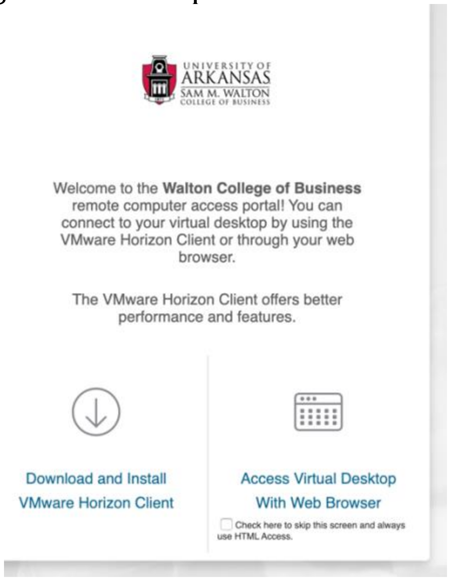
Below are three additional queries that can be produced by the database. Our team has proposed these queries based on data that we believe would be helpful to the operations of Assistance League. Further, we have provided an explanation for each query to provide our client with an understanding of how these may help Assistance League.

Query #	Question	Why?	SQL	Partial Output	Recap of Findings
1	How many of each item do we have on-hand?	This helps OSB monitor their current inventory levels.	SELECT P.PName, P.PQty_OH FROM TProduct P join TSaleOrderLine SOL on P.ProductID = SOL.SOLProductID join TSaleOrder SO on SOL.SOLSOID = SO.SOID	 PName PQty_OH 1 Dress Shoes 44 2 Short-Sleeve Shirt 50 3 Socks 50 4 Underwear 82 5 Courdoroys 87 6 T-Shirt 96 7 Khakis 141 8 Coat 182 9 Jeans 191 10 Sneakers 201 11 Long-Sleeve Shirt 228	It appears that we have plenty of tops/coats and may need more pant items.
2	How many students were male and female?	This allows us to see which products we should have more on hand of based on gender.	SELECT DISTINCT COUNT(St.StGenderID) as Gender_Count, G.GenderName FROM TGender G join TStudent St on G.GenderID = St.StGenderID	 Gender_Count GenderName 1 736 Female 2 864 Male	This reveals that there are slightly more boys than girls that visit OSB.
3	What brand of clothing was most purchased by students?	Helps the client choose brands that students like the most and eliminate too much unwanted inventory in the store.	SELECT B.BrandName, COUNT(SOL.SOLSOID) as SOL_Count FROM TBrand B join TProduct P on B.BrandID = P.PBrandID join TSaleOrderLine on SOL.SOLProductID = P.ProductID GROUP BY B.BrandName ORDER BY COUNT(SOL.SOLSOID) Desc	 BrandName SOL_Count 1 Cotton On 17 2 Reebok 14 3 Levi's 11 4 PUMA 11 5 New Balance 10 6 Nike 8 7 The Children's Place 8 8 Forever21 7 9 Gap 7 10 Hanes 7	Cotton On is the most purchased brand.

User Documentation

Below, our team has outlined detailed steps accompanied by visuals that will be needed for users to access the database. The process starts within the University of Arkansas website and users will be guided through all steps in order to view the database diagram along with tables and stored procedures.

1. Follow the link to the University of Arkansas website and click on the "Access Virtual Desktop Through Browser" option.



UNIVERSITY OF ARKANSAS SAM M. WALTON COLLEGE OF BUSINESS

Welcome to the **Walton College of Business** remote computer access portal! You can connect to your virtual desktop by using the VMware Horizon Client or through your web browser.

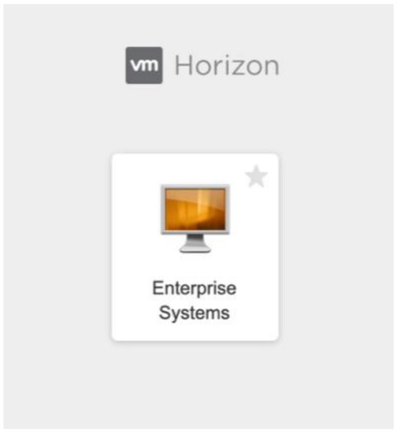
The VMware Horizon Client offers better performance and features.

Download and Install VMware Horizon Client

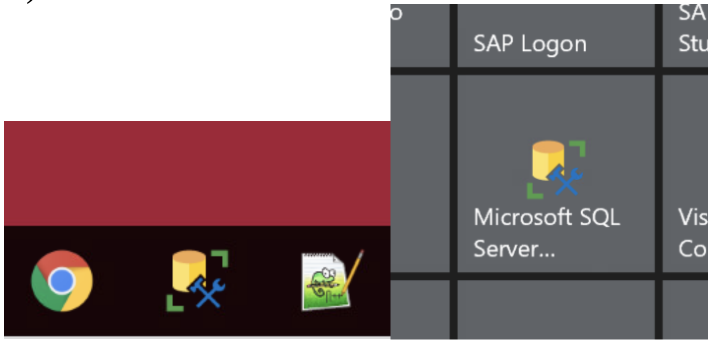
Access Virtual Desktop With Web Browser

Check here to skip this screen and always use HTML Access.

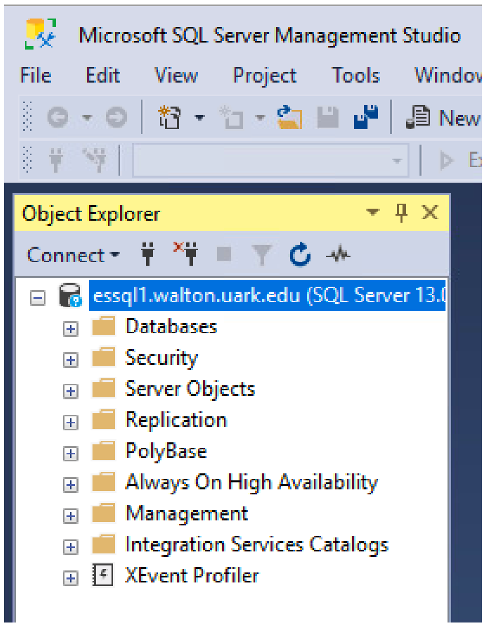
2. Log into the University of Arkansas website using given login credentials to access the VMware Client.
3. Click "Accept".
4. Select the "Enterprise Systems" icon, this will open a login screen to a Windows desktop.



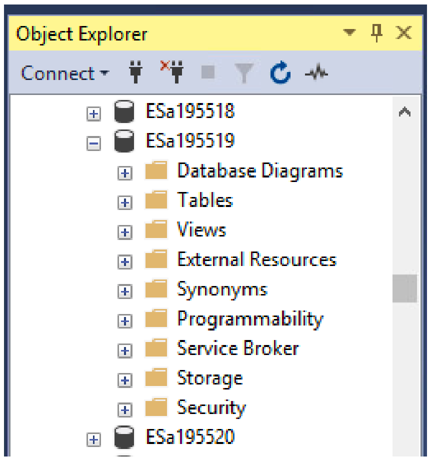
5. Once logged into the desktop, search for the program "Microsoft SQL Server Management Studio" and open the program (also found on the taskbar next to Google Chrome).



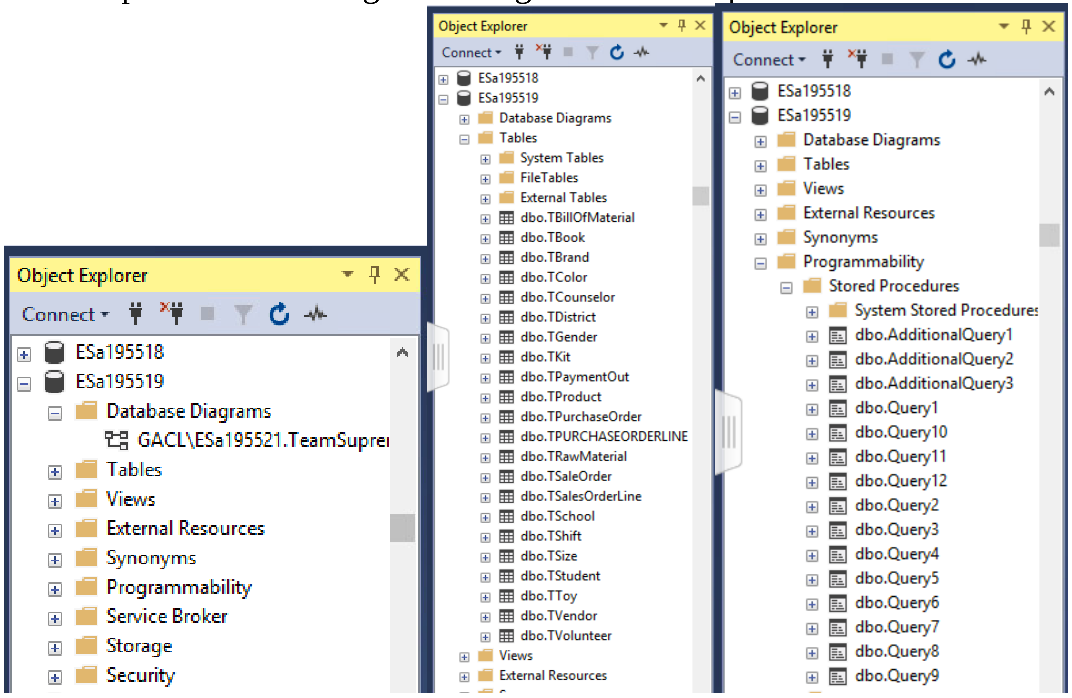
6. You will be prompted to Connect to the Server. To do so, input "essql.walton.uark.edu" into the Server Name box. Use given username and password credentials and log into the server.
7. Once logged in, navigate the Object Explorer to find the given database. It will be under the ID of "Esa195519".


Microsoft SQL Server Management Studio
File
Edit
View
Project
Tools
New
Object Explorer
Connect
essql1.walton.uark.edu (SQL Server 13.
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
XEvent Profiler

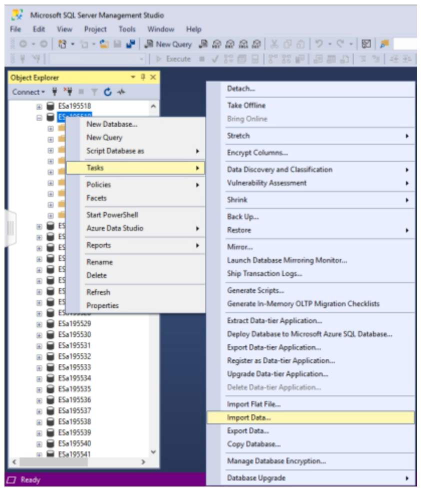
8. From here, you can access the created Database Diagram, Tables, and Programmability (to view stored queries).



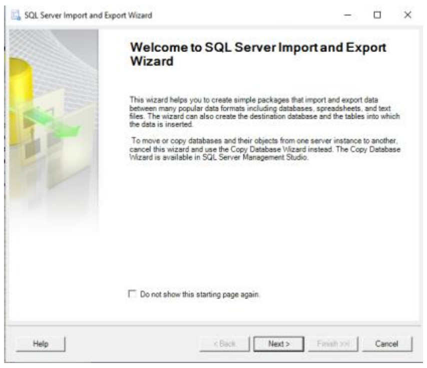
9. Use the plus and minus signs to navigate to various parts of the database.



10. To enter new data into the database, right click on the database number – ESa195519. Scroll down to highlight “Tasks” and then scroll to the right to highlight and click “Import Data”.


Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
Object Explorer
New Database...
New Query
Script Database as
Tasks
Policies
Facets
Start PowerShell
Azure Data Studio
Reports
Rename
Delete
Refresh
Properties
Detach...
Take Offline
Bring Online
Stretch
Encrypt Columns...
Data Discovery and Classification
Vulnerability Assessment
Shrink
Back Up...
Restore
Mirror...
Launch Database Mirroring Monitor...
Ship Transaction Logs...
Generate Scripts...
Generate In-Memory OLTP Migration Checklists
Extract Data-tier Application...
Deploy Database to Microsoft Azure SQL Database...
Export Data-tier Application...
Register as Data-tier Application...
Upgrade Data-tier Application...
Delete Data-tier Application...
Import Flat File...
Import Data...
Export Data...
Copy Database...
Manage Database Encryption...
Database Upgrade
Ready

11. Next, you will see the Import Wizard. Click “Next”.

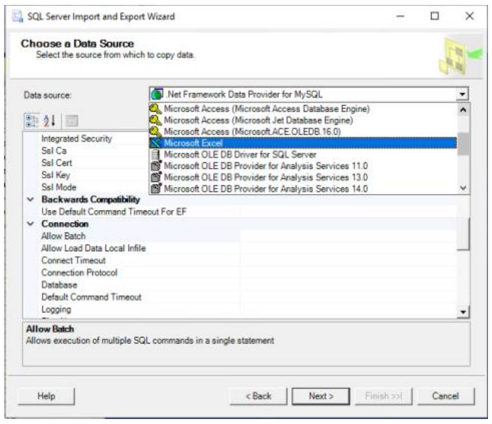


Welcome to SQL Server Import and Export Wizard

This wizard helps you to create simple packages that import and export data between many popular data formats including databases, spreadsheets, and text files. The wizard can also create the destination database and the tables into which the data is inserted.

To move or copy databases and their objects from one server instance to another, cancel this wizard and use the Copy Database Wizard instead. The Copy Database Wizard is available in SQL Server Management Studio.

12. You will have to choose a Data Source to import your data from. Choose Microsoft Excel if your data is stored in an Excel file. Click “Next”.


SQL Server Import and Export Wizard

Choose a Data Source
Select the source from which to copy data.

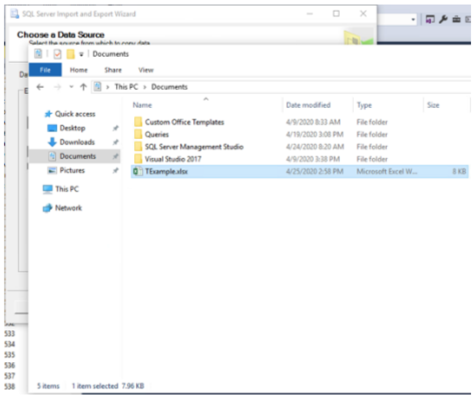
Data source: .Net Framework Data Provider for MySQL

Microsoft Access (Microsoft Access Database Engine)
Microsoft Access (Microsoft Jet Database Engine)
Microsoft Access (Microsoft.ACE.OLEDB.16.0)
Microsoft Excel
Microsoft OLE DB Driver for SQL Server
Microsoft OLE DB Provider for Analysis Services 11.0
Microsoft OLE DB Provider for Analysis Services 13.0
Microsoft OLE DB Provider for Analysis Services 14.0
Integrated Security
Ssl Ca
Ssl Cert
Ssl Key
Ssl Mode
Backwards Compatibility
Use Default Command Timeout For EF
Connection
Allow Batch
Allow Load Data Local Infile
Connect Timeout
Connection Protocol
Database
Default Command Timeout
Logging

Allow Batch
Allows execution of multiple SQL commands in a single statement

Help
< Back
Next >
Finish >>|
Cancel

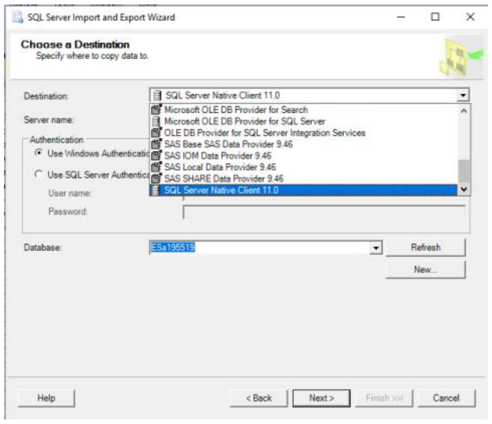
13. You will be prompted to choose an Excel file. Click “Browse” and find the file you would like to import.



Name	Date modified	Type	Size
Custom Office Templates	4/9/2020 8:33 AM	File folder	
Queries	4/19/2020 3:08 PM	File folder	
SQL Server Management Studio	4/24/2020 8:20 AM	File folder	
Visual Studio 2017	4/9/2020 3:38 PM	File folder	
TExample.xlsx	4/25/2020 2:58 PM	Microsoft Excel W...	8 KB

5 items 1 item selected 7.96 KB

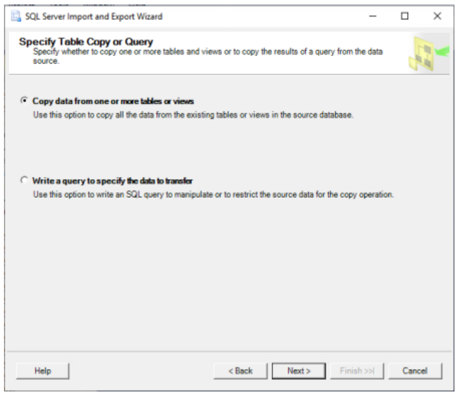
14. Next, you will have to choose a Destination Source. Find “SQL Server Native Client 11.0” in the drop-down menu. Be sure that you have the correct database chosen in the “Database:” drop-down box. Click “Next”.


SQL Server Import and Export Wizard
Choose a Destination
Specify where to copy data to.
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
OLE DB Provider for SQL Server Integration Services
SAS Base SAS Data Provider 9.46
SAS IOM Data Provider 9.46
SAS Local Data Provider 9.46
SAS SHARE Data Provider 9.46
SQL Server Native Client 11.0
Server name:
Authentication
Use Windows Authenticatio
Use SQL Server Authentic
User name:
Password:
Refresh
New...
Help
< Back
Next >
Finish >>|
Cancel

Destination: SQL Server Native Client 11.0

Database: ESa195519

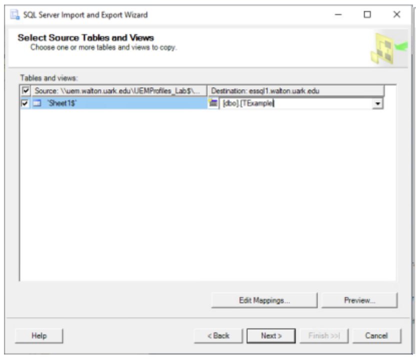
15. This screen will appear. Choose the top option: “Copy data from one or more tables or views”. Click “Next”.


SQL Server Import and Export Wizard
Specify Table Copy or Query
Help
< Back
Next >
Finish >>|
Cancel

Specify whether to copy one or more tables and views or to copy the results of a query from the data source.

- [✓ Copy data from one or more tables or views]
  Use this option to copy all the data from the existing tables or views in the source database.
- Write a query to specify the data to transfer
  Use this option to write an SQL query to manipulate or to restrict the source data for the copy operation.

16. At this point, you can rename the table appropriately by double clicking the right-hand box. This table has been renamed as "TExample". Once you have renamed the table, click the "Edit Mappings" button. Click "Finish" twice.

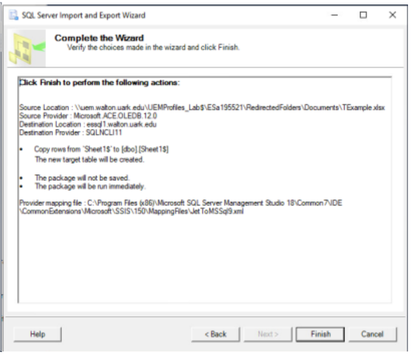


SQL Server Import and Export Wizard

Select Source Tables and Views
Choose one or more tables and views to copy.

Tables and views:

✓ Source: \\uem.walton.uark.edu\UEMProfiles_Lab$\...	Destination: essql1.walton.uark.edu
✓ 'Sheet1$'	[dbo].[TExample]



SQL Server Import and Export Wizard

Complete the Wizard
Verify the choices made in the wizard and click Finish.

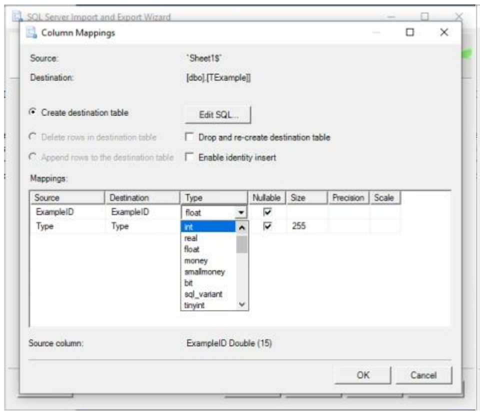
Click Finish to perform the following actions:

Source Location : \\uem.walton.uark.edu\UEMProfiles_Lab$\ESa195521\RedirectedFolders\Documents\TExample.xlsx
Source Provider : Microsoft.ACE.OLEDB.12.0
Destination Location : essql1.walton.uark.edu
Destination Provider : SQLNCLI11

- Copy rows from 'Sheet1$' to [dbo].[Sheet1$]
  The new target table will be created.
- The package will not be saved.
- The package will be run immediately.

Provider mapping file : C:\Program Files (x86)\Microsoft SQL Server Management Studio 18\Common7\IDE \CommonExtensions\Microsoft\SSIS\150\MappingFiles\JetToMSSql9.xml

17. Use the drop-down option in the "Type" column to adjust the data types of columns. Here, "ExampleID" is being set as an integer. The size of varchar and nvarchar data types can also be adjusted as shown below in the pane on the right. Once these values are adjusted, click "OK" and then "Next".



Column Mappings

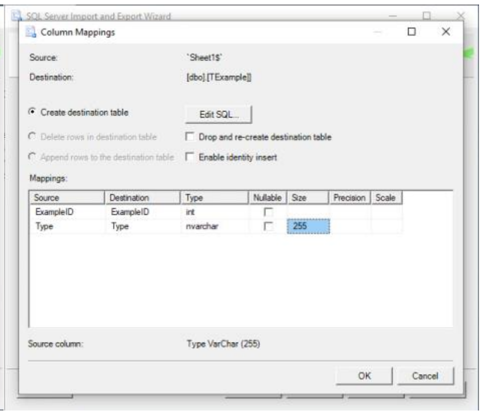
Source: 'Sheet1$'
Destination: [dbo].[TExample]]

(●) Create destination table
( ) Delete rows in destination table
( ) Append rows to the destination table
☐ Drop and re-create destination table
☐ Enable identity insert

Mappings:

Source	Destination	Type	Nullable	Size	Precision	Scale
ExampleID	ExampleID	float	✓			
Type	Type	int	✓	255		

Source column: ExampleID Double (15)



Column Mappings

Source: 'Sheet1$'
Destination: [dbo].[TExample]]

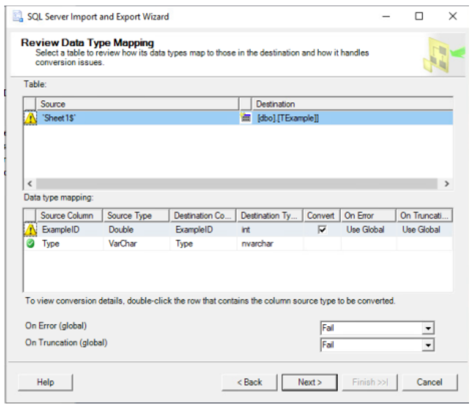
(●) Create destination table
( ) Delete rows in destination table
( ) Append rows to the destination table
☐ Drop and re-create destination table
☐ Enable identity insert

Mappings:

Source	Destination	Type	Nullable	Size	Precision	Scale
ExampleID	ExampleID	int	☐			
Type	Type	nvarchar	☐	255		

Source column: Type VarChar (255)

18. The next screen will display any errors before the data is copied to its destination. In this case, the error will not affect the implementation, so click "Next".



SQL Server Import and Export Wizard

Review Data Type Mapping
Select a table to review how its data types map to those in the destination and how it handles conversion issues.

Table:

	Source		Destination
⚠	'Sheet1$'		[dbo].[TExample]]

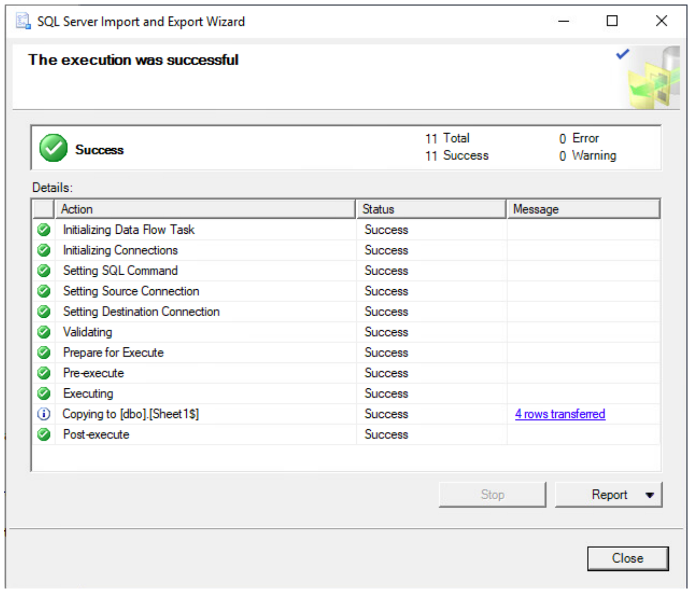
Data type mapping:

	Source Column	Source Type	Destination Co...	Destination Ty...	Convert	On Error	On Truncati...
⚠	ExampleID	Double	ExampleID	int	✓	Use Global	Use Global
●	Type	VarChar	Type	nvarchar			

To view conversion details, double-click the row that contains the column source type to be converted.

On Error (global): Fail
On Truncation (global): Fail

19. The following pane will appear and it will display the number of rows that were successfully transferred. Press “Close”.



SQL Server Import and Export Wizard

The execution was successful

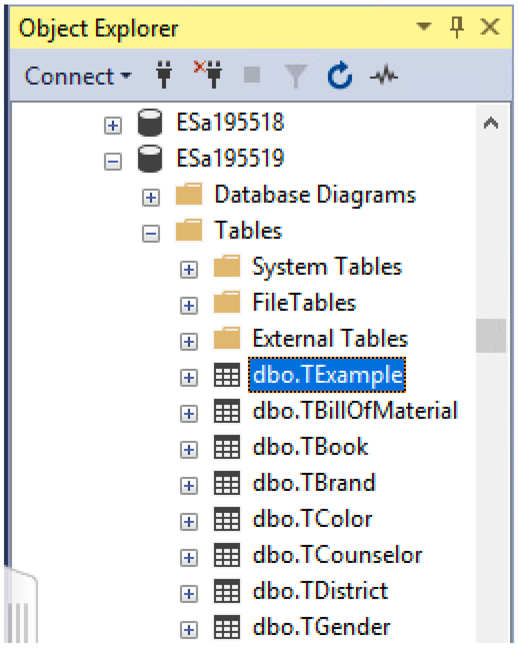
Success

11 Total 0 Error
11 Success 0 Warning

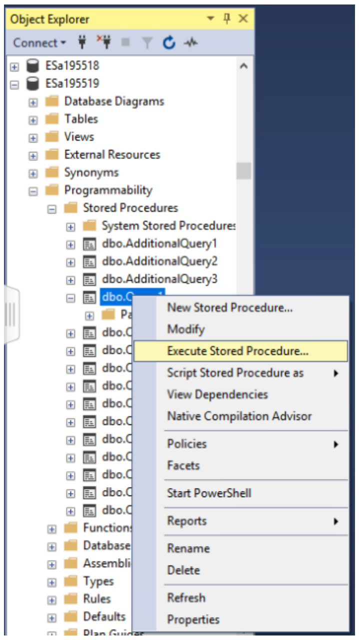
Details:

Action	Status	Message
Initializing Data Flow Task	Success	
Initializing Connections	Success	
Setting SQL Command	Success	
Setting Source Connection	Success	
Setting Destination Connection	Success	
Validating	Success	
Prepare for Execute	Success	
Pre-execute	Success	
Executing	Success	
Copying to [dbo].[Sheet1$]	Success	4 rows transferred
Post-execute	Success	

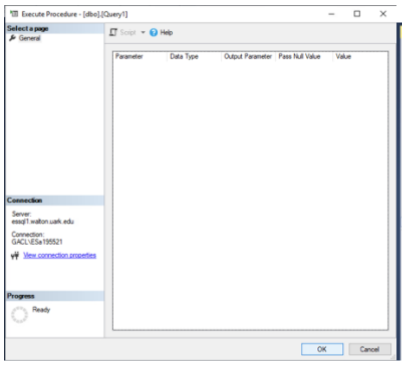
20. Refresh the database by right clicking “ESa195519” and clicking “Refresh”. The table will now appear in the “Tables” section.

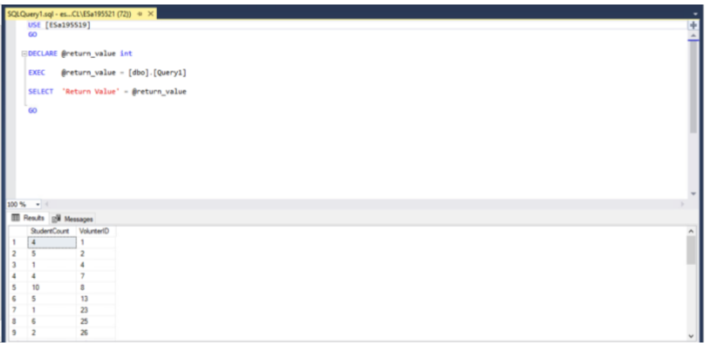


21. By opening the Stored Procedures found under the Programmability tab (screenshot shown above on far right), you can execute stored queries. Right click on the query you would like to execute and scroll across to the list to highlight “Execute Stored Procedure”.



22. Next, a box will appear. Click “OK” to execute the query. The query will automatically execute and the screen on the right will appear. The results of the query are shown under the “Results” tab on the lower half of the screen. Simply click the “X” on the highlighted tab or press Ctrl 4 to exit out of the query pane.





```
USE [ESa195519]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[Query1]

SELECT  'Return Value' = @return_value

GO
```

	StudentCount	VolunterID
1	4	1
2	5	2
3	1	4
4	4	7
5	10	8
6	5	13
7	1	23
8	6	25
9	2	26

What We Learned Throughout This Process

This entire project has been an incredibly beneficial process. It has required vast amounts of communication and hard work. From the client interview to the hands-on experience with a database application, our group has worked together and learned so much about databases.

There have been some incredible learning opportunities for the group that have prepared us for our careers. Firstly, we learned how to work as a team. This project demanded a lot of communication and leadership skills to be successful. This project has been especially hard since our team was forced into only working online because of the current global pandemic. We have utilized multiple communication channels: group me, one drive, teams, and zoom. This has taught us the importance of online and virtual group work. As the business world is adapting to technological advancements, it is paramount that workers know how to use these team channels. Secondly, the group has gotten hands-on experience with database design and implementation. Being MIS majors, this has been crucial to our skills and knowledge in the field.

Overall, the group has had a wonderful experience helping operation school bell meet its database needs. We have overcome multiple challenges, and we believe our work will benefit our client moving forward.

Member:	What we learned:
Chase Bonar	I have learned how important online communication and group work is. The current global situation has put our team in a unique spot. It required me to use tools I have never used before.
Talia Dean	I have learned that there are several ways to model data and satisfy client requirements. It has been insightful to listen to the methods and thought processes that my team members use and adapt my way of thinking from theirs.
Armin Garcia	The most important lesson learned is constant communication and listening to each member input to implement an efficient design. Moreover, working online rather than in-person presented itself as a challenge. Especially working on the SQL server, our group experienced constant technical failures that resulted in losing time having to redo the tables, diagrams, and queries.
Ryan Henry	I have learned how important the communication between group members and our professor are. In this unique situation, I think communicating, delegating tasks, and flexibility are the most important parts of the completion of this project.
Charlie Robinson	I have learned the importance of being consistent when working on a team project like this. It was challenging to stay focused and committed to this project as well as everything else going on outside of this class.

Appendix

Team Contract

To ensure clear communication among our team members throughout the course of this project, we developed a team contract to adhere to. Below we have listed our contact information and availability as a reference point for one another. We have highlighted our strengths as a team as well as the strengths of our individual members in order to effectively delegate tasks.

Team Members:

Name	Email	Phone	Strengths	Availability
Chase Bonar	Chasesb@ou.edu	(817) 913-5324	Innovative	MWF before 4pm
Talia Dean	Taliajdean@ou.edu	(469) 307-6043	Organized Punctual	MWF
Armin Garcia	Armingarcia@ou.edu	(832) 656-9176	Creative Analytical Detail-Oriented	MWF Evenings T/TH Mornings
Ryan Henry	Ryan.henry@ou.edu	(918) 407-2830	Organization Late night deadlines	MWF 2-8pm T/TH 9-1:30pm; 4-8pm
Charles Robinson	Charlie@ou.edu	(405) 626-8494	Listening Organization	Any day except T/TH

Unique Capabilities: Highly Organized
Team Expectations (for Peer Evaluation): Each member does their part to help us succeed.
Team Motto: "If you ain't first, you're last."

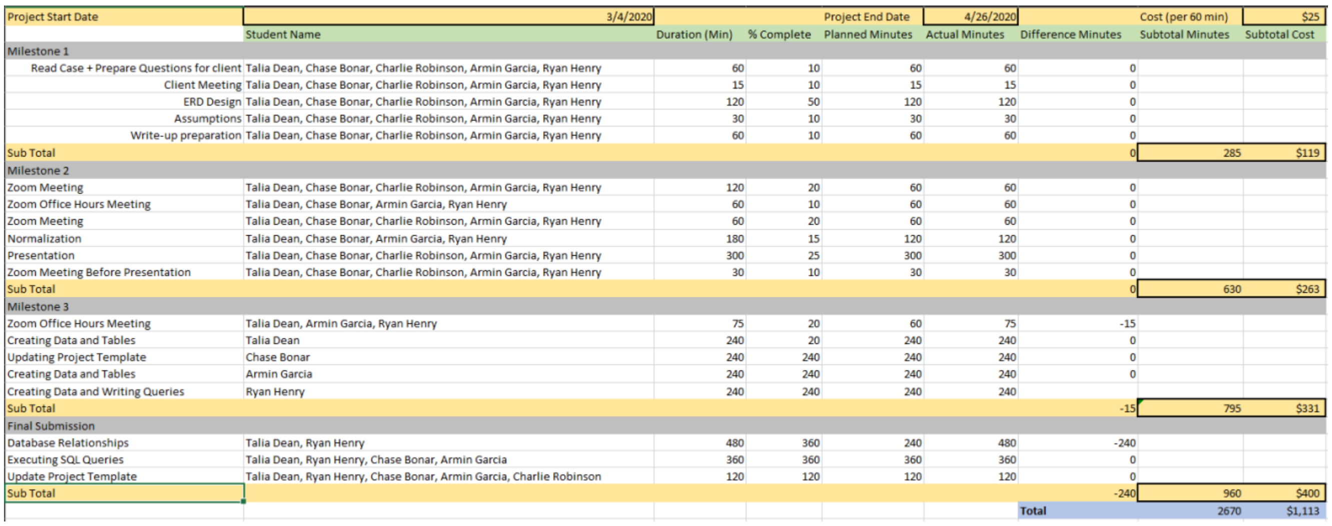
Data Dictionary Model

Table	Key	Field Name	Data Type	Size	Null	References	Sample
TSize	PK	SizeID	Int		Not Null		1
TColor	PK	ColorID	Int		Not Null		1
		ColorName	Varchar		Not Null		Red
TGender	PK	GenderID	Int		Not Null		1
		GenderName	Varchar		Not Null		Female
TBrand	PK	BrandID			Not Null		1
		BrandName			Not Null		Adidas
TKit	PK	KitID	Int(Auto-increment)		Not Null		1
		RMComponentID	Int		Not Null	TRawMaterial	1
TVendor	PK	VendorID	Int(Auto-increment)		Not Null		1
	FK	VZipCode	Varchar	5	Not Null	TZip	73072
		VFirstName	Varchar	50	Not Null		James
		VLastName	Varchar	50	Not Null		Smith
		VStreet	Varchar	100	Not Null		123 Las Vegas Blvd.
		VPhone	Varchar	10	Not Null		2144089276
TRawMaterial	PK	RMID	Int(Auto-increment)		Not Null		1
	FK	RMVendorID	Int		Not Null	TVendor	1
		RMName	Varchar	50	Not Null		Toothpaste
		RMQty_OH	Int	100	Null Allowed		5
TBillofMaterial	PK	BOMID	Int(Auto-increment)		Not Null		1
	FK	BOMProductID	Int		Not Null	TProduct	1
	FK	BOMRMID	Int		Not Null	TRawMaterial	1
		RMQty	Int	100	Not Null		5
TProduct	PK	ProductID	Int(Auto-increment)		Not Null		1
	FK	PSizeID	Int		Not Null	TSize	1
	FK	PColorID	Int		Not Null	TColor	1
	FK	PGenderID	Int		Not Null	TGender	1
	FK	PBrandID	Int		Not Null	TBrand	1
		PName	Varchar	50	Not Null		Jeans
		PCategory	Varchar	20	Not Null		Pants
		PPurchaseCost	Varchar	4	Not Null		$30.05
		PQty_OH	Int	500	Null Allowed		100
		PQty_Avail	Int	500	Null Allowed		50
TShift	PK	ShiftID	Int(Auto-increment)		Not Null		1
	FK	SVolunteerID	Int		Not Null	TVolunteer	1
		SDate	Date		Not Null		4/13/2020
		SClockIn	Date/Time		Not Null		4/13/2020
		SClockOut	Date/Time		Not Null		4/13/2020
TVolunteer	PK	VolunteerID	Int(Auto-increment)		Not Null		1
	FK	VolZipCode	Varchar	5	Not Null	TZip	90210
		VolFName	Varchar	50	Not Null		Karen

		VolLName	Varchar	50	Not Null		Smith
		VolStreet	Varchar	100	Not Null		123 Hollywood Blvd.
		VolPhone	Varchar	10	Not Null		2144089276
TPaymentOut	PK	PayoutID	Int(Auto-increment)		Not Null		1
	FK	PVolunteer-ChairID	Int		Not Null	TVolunteer	1
		PPaymentOut	Decimal		Not Null		$300.50
		PPaymentDate	Date/Time		Not Null		4/13/2020 7:04pm
TPurchaseOrder	PK	POID	Int(Auto-increment)		Not Null		1
	FK	PurPayOutID	Int		Not Null	TPaymentOut	1
	FK	PurVendorID	Int		Not Null	TVendor	1
	FK	PurVolunteerID-Inspect	Int		Not Null	TVolunteer	1
	FK	PurVolunteerID-Receives	Int		Not Null	TVolunteer	1
		PurDescription	Varchar		Not Null		Order includes clothing, etc.
		PurPOAmount	Decimal		Not Null		$100.00
		PurTotal	Decimal		Not Null		$150.00
		PurDatePaid	Date		Not Null		4/13/2020
		PurDate	Date		Not Null		4/13/2020
TPurchaseOrderLine	PK	POLID	Int(Auto-increment)		Not Null		1
	FK	POLPOID	Int		Not Null	TPurchaseOrder	1
	FK	POLProductID	Int		Not Null	TProduct	1
		POLQty	Int		Not Null		50
		POLLineTotal	Decimal		Not Null		$400.00
TDistrict	PK	DistrictID	Int(Auto-increment)		Not Null		1
		DistrictName	Varchar		Not Null		Little Axe
TCounselor	PK	CounselorID	Int(Auto-increment)		Not Null		1
		CFName	Varchar		Not Null		Karen
		CLName	Varchar		Not Null		Smith
TSaleOrder	PK	SOID	Int(Auto-increment)	200	Not Null		1
	FK	SOSchoolID	Int		Not Null	TSchool	1
	FK	SOVolunteerID	Int		Not Null	TVolunteer	1
	FK	SOKitID	Int		Not Null	TKit	1
	FK	SOToyID	Int		Null Allowed	TToy	1
	FK	SOBookID	Int		Null Allowed	TBook	1
		SODate	Date/Time		Not Null		4/13/2020
		SOStudentAge	Int		Not Null		13
		SOGrade	Int		Not Null		5
		SOTotal	Decimal		Not Null		$50.00
TSchool	PK	SchoolID	Int(Auto-increment)		Not Null		1
	FK	SDistrictID	Int		Not Null	TDistrict	1
	FK	SCounselorID	Int		Not Null	TCounselor	1
		SSchoolName	Varchar		Not Null		Little Axe Elementary
		SStreet	Varchar		Not Null		123 ABC Dr.
	FK	SZipCode	Varchar		Not Null	TZip	73072

TStudent	PK	StudentID	Int(Auto-increment)		Not Null		1
	FK	StSchoolID	Int		Not Null	TSchool	1
	FK	StGenderID	Int		Not Null	TGender	1
		StFName	Varchar		Not Null		Bob
		StMName	Varchar		Not Null		Joe
		StLName	Varchar		Not Null		Smith
		StStreet	Varchar		Not Null		123 ABC Dr.
	FK	StZipCode	Varchar		Not Null	TZip	73072
		StDOB	Date		Not Null		4/13/2000
		StParentPhone	Varchar		Not Null		9726789435
		StFacultyRecommendation	Varchar		Not Null		Karen Smith
		StPrefferedColor	Varchar		Not Null		Pink
TSalesOrderLine	PK	SOLID	Int(Auto-increment)	450	Not Null		1
	FK	SOLSOID	Int		Not Null	TSaleOrder	1
	FK	SOLProductID	Int		Not Null	TProduct	1
		Qty	Int		Not Null		50
		SOLTotal	Decimal		Not Null		$45.00
TToy	PK	ToyID	Int(Auto-Increment)		Not Null		1
		TName	Varchar		Not Null		Football
		TQty	Int		Null Allowed		1
TBook	PK	BookID	Int(Auto-Increment)		Not Null		1
		BTitle	Varchar		Not Null		Harry Potter
		BQty	Int		Null Allowed		1

Project Management



Project Start Date	3/4/2020			Project End Date	4/26/2020		Cost (per 60 min)	$25
	Student Name	Duration (Min)	% Complete	Planned Minutes	Actual Minutes	Difference Minutes	Subtotal Minutes	Subtotal Cost
Milestone 1								
Read Case + Prepare Questions for client	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	60	10	60	60	0		
Client Meeting	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	15	10	15	15	0		
ERD Design	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	120	50	120	120	0		
Assumptions	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	30	10	30	30	0		
Write-up preparation	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	60	10	60	60	0		
Sub Total						0	285	$119
Milestone 2								
Zoom Meeting	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	120	20	60	60	0		
Zoom Office Hours Meeting	Talia Dean, Chase Bonar, Armin Garcia, Ryan Henry	60	10	60	60	0		
Zoom Meeting	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	60	20	60	60	0		
Normalization	Talia Dean, Chase Bonar, Armin Garcia, Ryan Henry	180	15	120	120	0		
Presentation	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	300	25	300	300	0		
Zoom Meeting Before Presentation	Talia Dean, Chase Bonar, Charlie Robinson, Armin Garcia, Ryan Henry	30	10	30	30	0		
Sub Total						0	630	$263
Milestone 3								
Zoom Office Hours Meeting	Talia Dean, Armin Garcia, Ryan Henry	75	20	60	75	-15		
Creating Data and Tables	Talia Dean	240	20	240	240	0		
Updating Project Template	Chase Bonar	240	240	240	240	0		
Creating Data and Tables	Armin Garcia	240	240	240	240	0		
Creating Data and Writing Queries	Ryan Henry	240	240	240	240			
Sub Total						-15	795	$331
Final Submission								
Database Relationships	Talia Dean, Ryan Henry	480	360	240	480	-240		
Executing SQL Queries	Talia Dean, Ryan Henry, Chase Bonar, Armin Garcia	360	360	360	360	0		
Update Project Template	Talia Dean, Ryan Henry, Chase Bonar, Armin Garcia, Charlie Robinson	120	120	120	120	0		
Sub Total						-240	960	$400
						Total	2670	$1,113