University of Oklahoma

Elysian Fly Company

The Strangers

Reed Brackett, Blake Garrett, Sara Hernandez, Bryston Stark, Coreyonn Stokes

MIS 3353 – Database Management
Professor Swetha Siripurapu
April 28, 2024

THE STRANGERS

DON'T BE A STRANGER

Executive Summary

Our team's focus is put onto everybody who will and might interact with our work because we do not want anybody to be a stranger. Creating and implementing a database can be a confusing process but our objective is to make everybody comfortable and familiar with every step to get there. Our team has put in countless hours to explain what we did and why we did it, so no questions are left unasked.

We started by analyzing the requirements and specifications made by my Elysian, asking the client about any uncertainties. Our analysts used this to create an Entity Relationship Diagram or an ERD which acted as a visual blueprint for the database. This is an easy way to view the foundation and allows the client to make any corrections to make our work as accurate as possible.

After lots of revisions to ensure the database can return all needed information and requests, we were ready to move to logical design. This process defined the constraints between relations as well as normalized them. The ERD was converted into normalized relations which allowed for the final creation of the database to be seamless and flawless.

Our team then created a data dictionary which contains all the tables names, fields, and the information needed to create the tables in the database. This was referenced to create the database in SQL Server Management Studio (SSMS) by Microsoft, which will be used to access it. This data dictionary will also be handed to the client to help familiarize the user and any other viewer with the newly built database. To avoid any confusion accessing it, a step-by-step guide to reach and interact with the database is shown here.

A project task and cost tracking form were created to accurately document everything that brought us to the finished product. This can be accessed here. Each member's name can be seen as well as what they were doing. Along with that is time spent and the portion completed by that member. Our agreed upon pay of $25/hour was used to calculate the subtotal pay (pretax) of $2,048. The Strangers thank you for this opportunity and are open to all assistance needed in the future or with Elysian's new database.

Table of Contents

Get to Know the Team: The Strangers

	Reed Brackett MIS Major Junior Internship Experience: N/A Background: Experience with handling large amounts of legal files and inputting documents into law firm's computer system.
	Blake Garrett MIS Major Junior Internship Experience: N/A Background: Detail-oriented student with experience working in team environments in both customer service and business operations
	Sara Hernandez MIS Major Junior Internship Experience: N/A Background: Experience in customer service as well as manufacturing processes.
	Bryston Stark MIS Major Sophomore Internship Experience: N/A Background: Several years of experience working with customers in retail, sales and entertainment.
	Coreyonn Stokes MIS Major Junior Internship Experience: N/A Background: Ambitious, goal-oriented student with experience working in retail and customer service

Conceptual Design

Conceptual design is what can be thought of as the blueprint for a database. It involves creating entities and attributes as well as establishing relationships between entities. The client gave our analyst specifications and certain requirements that need to be represented in the ERD. With this information, our team can create conceptual designs or blueprints. We do this by creating an entity relationship diagram or ERD. We do this in the web application LucidChart. This is easy to visualize, which allows the client to see the specifications and requirements being fulfilled. Entities are anything from people, places, products, or anything we want to keep track of. Essentially entities are nouns. In this case we need to track customers, flies, trips, employees, and much more. By creating these entities, we can track each by a unique primary key and all the attributes within each entity. Now for relationships between entities, they are read from left to right and right to left. Essentially relationships are verbs. The verb is usually a little different going the other direction. There can be many different types of relationships in an ERD such as one-to-one, zero-to-many, one-to-many, or many-to-many. It simply depends on how each entity relates to each other. For example, a customer could have zero-to-many purchases. This is a significant part of the process because it helps decide what tables are essential and which ones can be deleted. It is also easier to interact with the database in the future because it allows the user to visualize where information is and how it relates to each other.

The Client Meeting

We scheduled a client meeting with the owner of Elysian Fly Company to clear up anything that had not been clear with the case. Our whole team got together with the client at 3:00 p.m. on March 3 and asked the questions we had put together to better understand the company’s request.

- Meeting Time: 3:00 p.m. March 3, 2024
- Location: MIS-3353-002 Class Zoom
- Interviewers: Reed Brackett, Blake Garrett, Sara Hernandez, Bryston Stark, and Coreyonn Stokes
- Interviewee: Owner of Elysian Fly Company

Q&A During the Meeting & Information We Learned

Our team collected a pool of questions before the meeting to better understand what the client was requesting. We recorded the questions down along with the answers we received. These questions helped shape what the foundation of our database would be.

Q1 - Can groups be combined during guided trips?
Yes.

Q2 - Can a guide be classified as an employee role?
Not necessarily, they are tracked differently.

Q3 - Can accounting employees be AR or AP as well as general and purchasing or does the separation only exist between AR and AP?
Accounting employees are accounting employees.

Q4 - Are returns offered?
No.

Q5 – The case implies there are 4 customer types but only 3 are discussed, what is the correct amount?
There are only 3 customer types.

Q6 – What specific information needs to be tracked regarding guided fishing trips?
We need to know the numbers of customers in a reservation and the guides.

Q7 - How do you handle sales transactions, both in-person and phone orders? Are there any specific details related to capturing sales data that we should consider in designing the database?
In-person sales does not require customer documentation, online sales dealt strictly by 3rd party.

Q8 - Is every product unique because hand tying can cause variations in the product? Or are they categorized based on the vendor and the vendor is what separates the same product?
Each fly is a different product but not every product is unique based on vendor, pattern, color, size, and combinations.

Q9 – Are bundles unique products comprised of many different products?
Comprised of multiple products for a given amount of time.

Significant Assumptions

Our team did a great job clearing up misunderstandings within the client meeting, but with our limited time we were left to make a few significant assumptions down the line to help build our ERD. This was essential to make progress towards the common goal of creating an efficient and effective database.

1. Flies do not have a unique name but are rather explained by their specific attributes such as color, pattern, and size
2. A sales order can have multiple discounts applied
3. Carrier type is not tracked for delivery in or out because tracking data is not being held within the database
4. An employee is trained on one and only one role because the smaller size of the company
5. Database does not need to track trail information such as dates and regulations as the guide employee will know this information as these things are subject to change overtime

What is an ERD? Why is it necessary?

ERD stands for Entity Relationship Diagram. It is a useful way to visualize how different things interact with one another in a database. The ERD lays down the blueprint we will use to structure the database while using phrases that are easily understandable such as a customer can make zero to many sales orders, or a guide can lead one to many groups. This allows us to illustrate what we are doing beforehand in a simpler and less expensive way. This conceptual database design allows our client to easily understand what the database's design is with no prior technical knowledge.

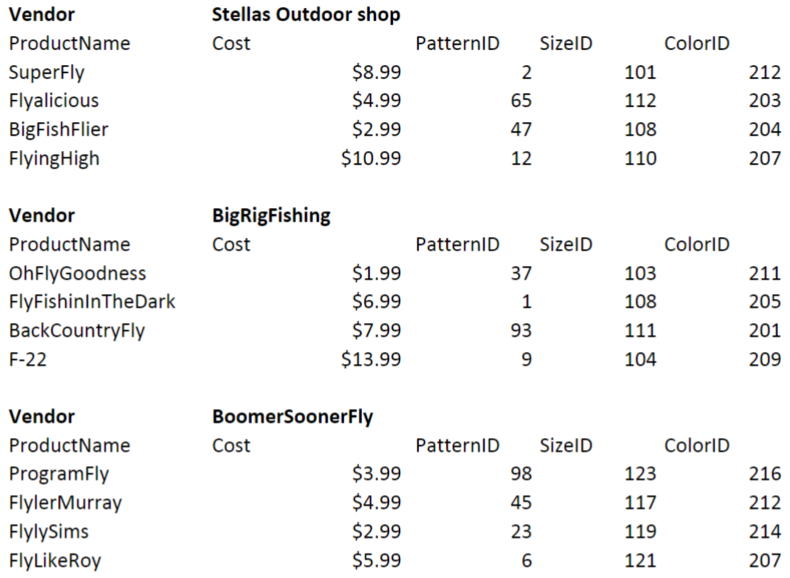
Business Cycles Used

What are business cycles? There are three generic business cycles that our team used as a foundation in creating our ERD. These include the revenue cycle, expenditure cycle, and production cycle. The revenue cycle is made whenever a company generates revenue. In its essence, the revenue cycle is when a customer creates a sales order to receive some form of service or product which is processed by an employee. The expenditure cycle is created when companies spend money. This is an important cycle that makes it easier to track ordering, receiving products and paying invoices. The third cycle is the production cycle which comes into play whenever a company produces anything. This allows easier tracking of product design and manufacturing.

Our team used the revenue, expenditure, and production business cycles. Elysian Fly Company offers products and services requiring the revenue cycle. The company purchases flies from multiple vendors requiring the expenditure cycle. Materials are required for flies tied in house and DIY kits. Both activities involve using raw materials, which led us to the production cycle.

Data Provided by Elysian

With the provided data below, we were able to see what vendors provide which products and all the specifications of said products. The specifications of the data include the vendor, vendor name, product name, cost, pattern identification, size identification, and color identification. The tables we created to house this data are the Vendor, Fly, Color, Pattern, and Size tables. We use these tables to see which vendors provide products based on the unique pattern, color, and size. In the Fly table, the three components are foreign keys because each fly is unique and not one fly is the same. Lastly, the cost of the flies is provided by the Fly table. The Vendor also provides a foreign key to the Fly table to show which vendor provides which flies. In addition, we had to normalize the newly created tables so the structural integrity of the database would be maintained.



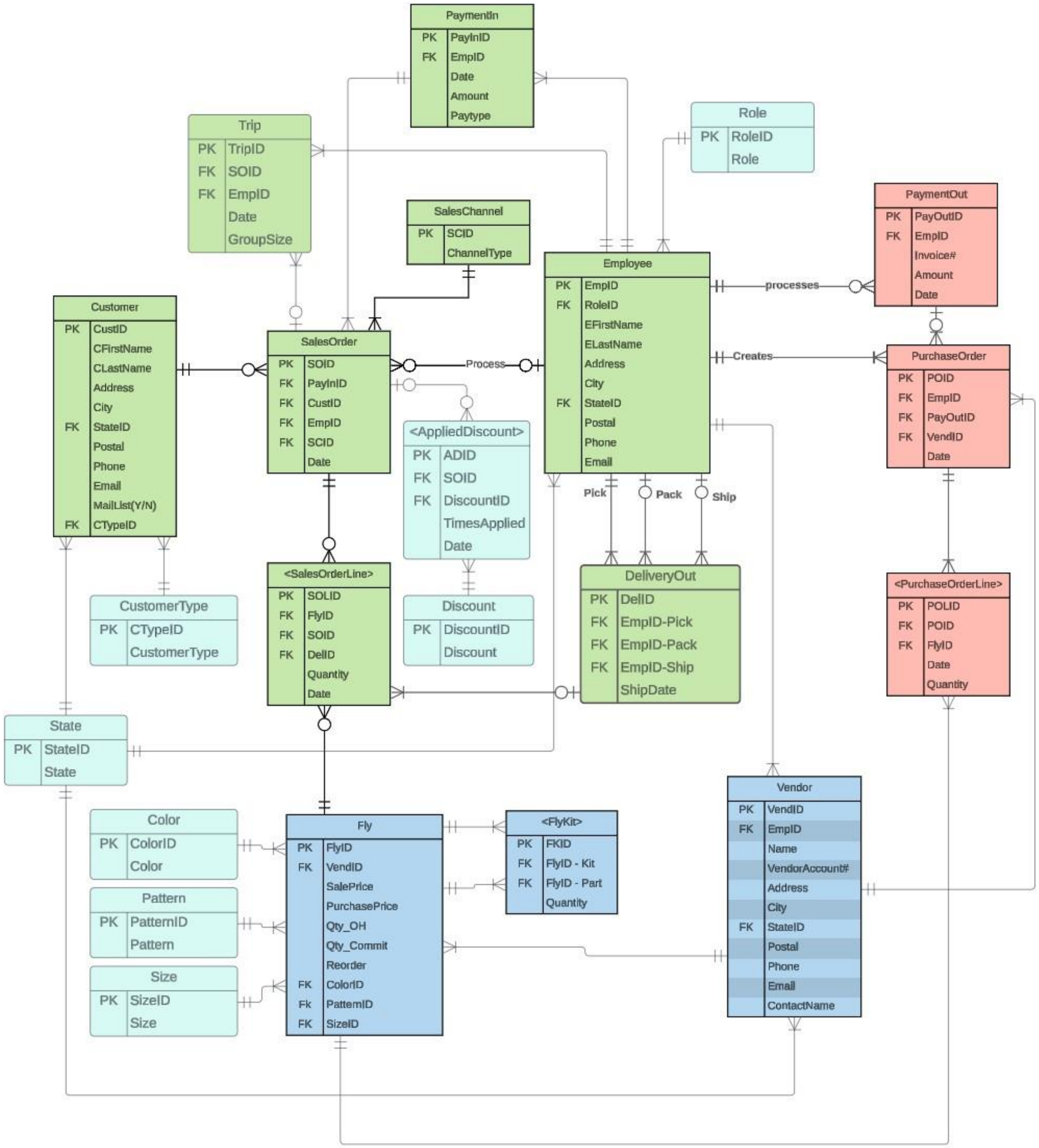
Vendor	**Stellas Outdoor shop**			
ProductName	Cost	PatternID	SizeID	ColorID
SuperFly	$8.99	2	101	212
Flyalicious	$4.99	65	112	203
BigFishFlier	$2.99	47	108	204
FlyingHigh	$10.99	12	110	207

Vendor	**BigRigFishing**			
ProductName	Cost	PatternID	SizeID	ColorID
OhFlyGoodness	$1.99	37	103	211
FlyFishinInTheDark	$6.99	1	108	205
BackCountryFly	$7.99	93	111	201
F-22	$13.99	9	104	209

Vendor	**BoomerSoonerFly**			
ProductName	Cost	PatternID	SizeID	ColorID
ProgramFly	$3.99	98	123	216
FlylerMurray	$4.99	45	117	212
FlylySims	$2.99	23	119	214
FlyLikeRoy	$5.99	6	121	207

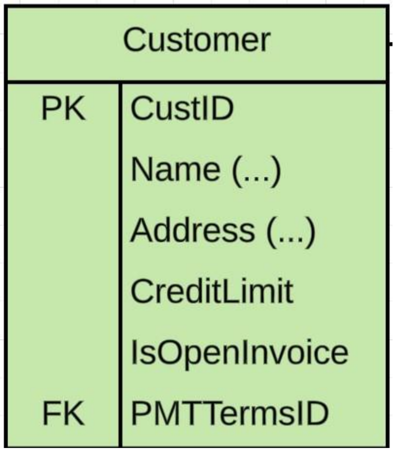
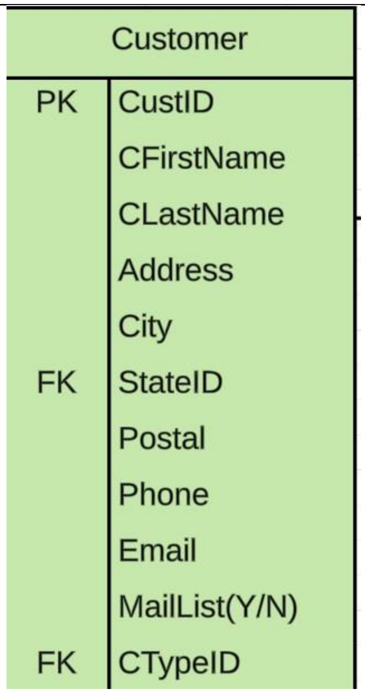
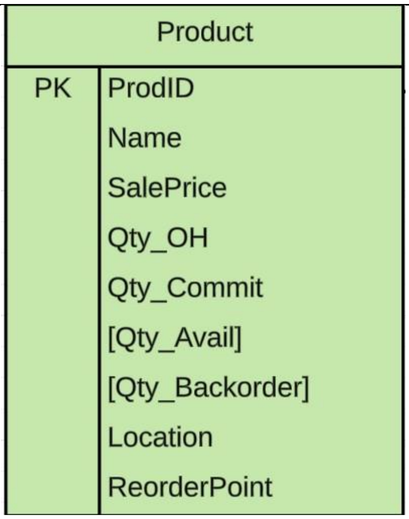
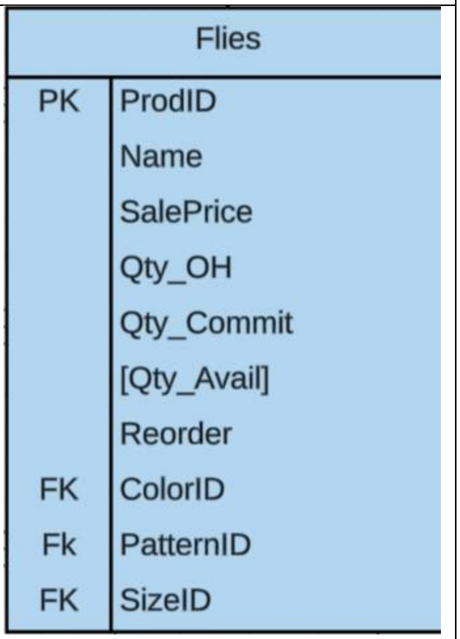
ERD Created

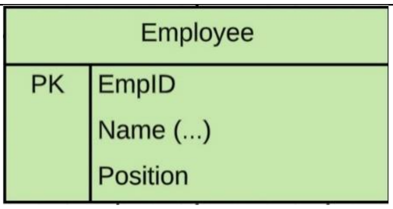
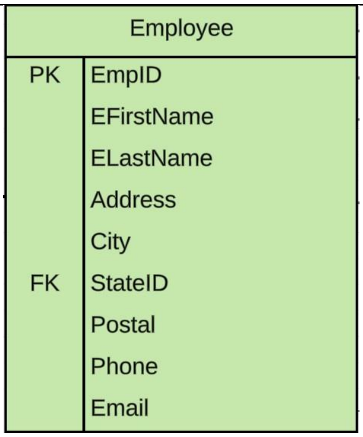
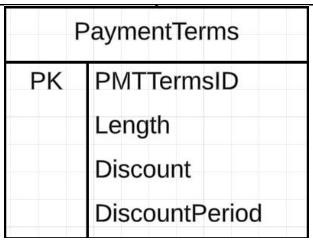
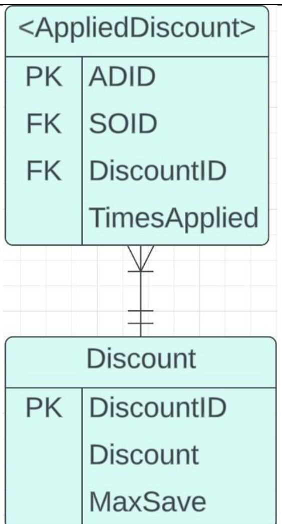
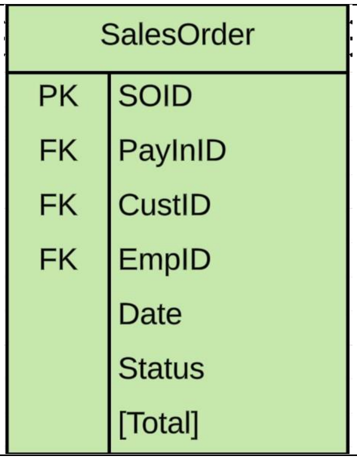
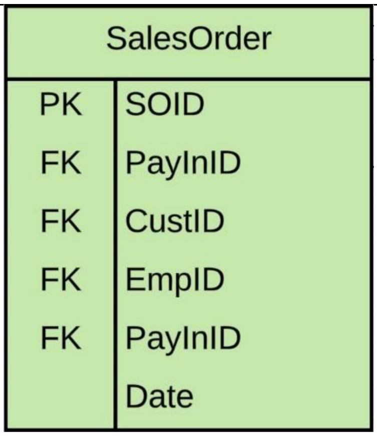
Our team worked collaboratively to come up with an ERD that sufficiently addresses all of Elysian Fly Company's requirements for a database. This was a long process that underwent many changes throughout its construction. This is largely due to the flexibility and many interpretations an ERD allows.

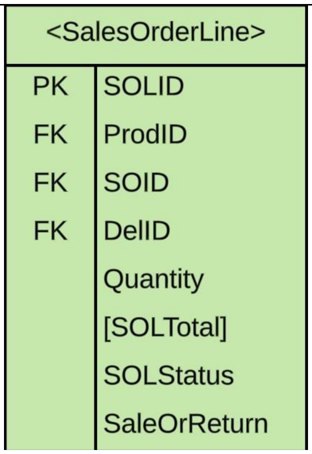
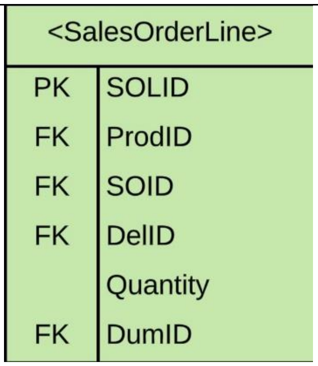
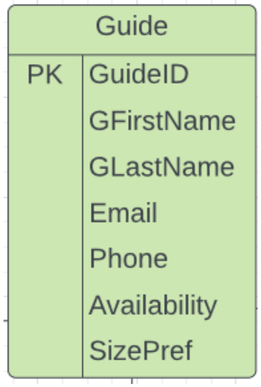
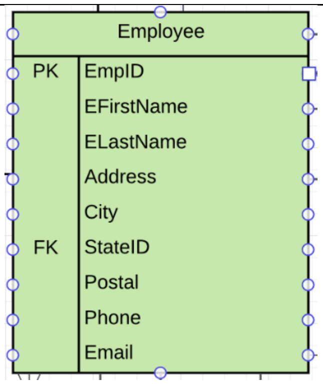


Click here for Lucid Chart

Changes made to generic ERDs

Change #	Original ERD	Updated ERD
Change #1 Our updated ERD better describes the customer with more specifics. One of the biggest changes is the reference table which states what the customer type is. This allows customers to be better tracked per request of Elysian Fly company.	 Customer PK CustID Name (...) Address (...) CreditLimit IsOpenInvoice FK PMTTermsID	 Customer PK CustID CFirstName CLastName Address City FK StateID Postal Phone Email MailList(Y/N) FK CTypeID
Change #2 The generic product table does not do a good job of describing the product. The flies consist of many different important combinations which our updated ERD does a great job of doing. Each of these describing attributes are done with reference tables as specific amounts were shared.	 Product PK ProdID Name SalePrice Qty_OH Qty_Commit [Qty_Avail] [Qty_Backorder] Location ReorderPoint	 Flies PK ProdID Name SalePrice Qty_OH Qty_Commit [Qty_Avail] Reorder FK ColorID Fk PatternID FK SizeID

Change #3 Our updated employee table provides more specifics than the general does. It cannot be seen in the screenshot but an association with the role table records what roles employees have been trained in and the date learned.	 Employee PK EmpID Name (...) Position	 Employee PK EmpID EFirstName ELastName Address City FK StateID Postal Phone Email
Change #4 Elysian Fly Company requested discount control and other various details regarding such. Our updated ERD relates the discount to the sale rather than with the customer. It also allows for the times a discount is applied to be tracked.	 PaymentTerms PK PMTTermsID Length Discount DiscountPeriod	 <AppliedDiscount> PK ADID FK SOID FK DiscountID TimesApplied Discount PK DiscountID Discount MaxSave
Change #5 The generic sales order tables have a calculated field which is not optimal. Our updated ERD removes this along with the status field. Payment information is also given in relationship to sales orders.	 SalesOrder PK SOID FK PayInID FK CustID FK EmpID Date Status [Total]	 SalesOrder PK SOID FK PayInID FK CustID FK EmpID FK PayInID Date

Change #6 In the client meeting we learned that Elysian Fly Co. does not accept returns. This is something the generic ERD models which we had to change in the updated ERD. This is also where we decided to relate the Dummy order which allows for the quantity to be tracked so inventory can be properly updated.	 <SalesOrderLine> PK SOLID FK ProdID FK SOID FK DelID Quantity [SOLTotal] SOLStatus SaleOrReturn	 <SalesOrderLine> PK SOLID FK ProdID FK SOID FK DelID Quantity FK DumID
Change #7 During the presentations, we learned that Guide does not need a separate table. They are employees. We also deleted the associative entities and reference tables pertaining to the now deleted Guide table.	 Guide PK GuideID GFirstName GLastName Email Phone Availability SizePref	 Employee PK EmpID EFirstName ELastName Address City FK StateID Postal Phone Email

Logical Design

The logical design phase of database development marks the transition from conceptual design to a structured blueprint ready for implementation. In this phase, the entity-relationship diagram (ERD) that was created in conceptual design phase is being dissected, with entities becoming tables, attributes transforming into columns, and relationships translating into foreign key constraints. With normalization, data redundancy is minimized, and data integrity is upheld by organizing the data into well-structured relations up to the specified normal form. This process essentially refines the schema, which optimizes the performance and ensures the usability of the database. Redundancies are eliminated, data types are defined, and constraints are enforced, which sets the stage for efficient data storage and retrieval. There is also documentation that captures these specifications, which provides a roadmap for the database's creation and maintenance, and it ensures the integrity and reliability of the database over time. In addition to the transformation of the ERD into relational structure and the application of normalized principles, the logical design phase involves further specification and documentation. This includes defining datatypes, constraints, and indexes for each table to enforce data integrity and optimize the query performance. Constraints such as primary keys and foreign keys are meticulously defined to maintain data consistency. Indexes are strategically implemented to expedite the retrieval of data. This is significant because the attention to detail ensures not only the structural integrity of the database but also its efficiency and scalability in handling large volumes of data of Elysian.

Normalization

Normalization is the process in which database developers ensure the database created is trustworthy and efficient. There are two steps to achieve normalization, to ensure atomicity of columns and to eliminate duplicates in the data. Atomicity breaks down each attribute so that it cannot be broken down any further. An example of this is a person's name. A person's name can be broken down by first, middle, last, and even go further into syllables. Secondly, deleting duplicates is significant because it causes many issues in the database regarding the records. However, if we created a database in which the tables are atomic, we should not have any duplicate issues. This is significant for the Elysian Fly Fishing database because this ensures that the tables are the most accurate. Normalization is also a process that should be done while creating a new database and remodeling one. If we were asked to add more tables, we would have to normalize the database again to ensure it is trustworthy. While going further in depth of normalization, we can discuss the three integrity constraints, being entity integrity, referential integrity, and domain integrity. Entity integrity emphasizes the need for every entity to have a primary key that is not null and does not change over time. Referential integrity can be defined as a relationship between entities, with said relationships a foreign key must match a valid primary key. Domain integrity emphasizes that all values in a field must be from the same domain, it allows us to split any attribute that is composite into smaller components and create an atomic attribute. Normal forms can also be discussed when talking about normalization. There are many types of normal forms but the main ones we discuss are 0NF, 1NF, 2NF, and 3NF. All these normal forms build on each other, getting more accurate and trustworthy. In conclusion, normalization is essential to building a database or remodeling one, the trustworthiness of a functioning business and all their data-related needs.

Normalization of Data Provided by Elysian

TPattern (PatternID, Pattern)

TSize (SizeID, Size)

TColor (ColorID, Color)

TVendor (VendorID, VendorName)

TFly (FlyID, ProductName, Cost, VendorID*, PatternID*, SizeID*, ColorID*)

Foreign Key VendorID references TVendor
Not Null
On Delete Restrict

Foreign Key PatternID references TPattern
Not Null
On Delete Restrict

Foreign Key SizeID references TSize
Not Null
On Delete Restrict

Foreign Key ColorID references TColor
Not Null
On Delete Restrict

Normalized Relations

TState (StateID, State)

TVendor (VendorID, VName, VendorAccount#, VAddress, VCity, VStateID*, VPostal, VPhone, VEmail, VContactName)

Foreign Key VStateID references TState
Not Null
On Delete Restrict

TColor (ColorID, Color)

TPattern (PatternID, Pattern)

TSize (SizeID, Size)

TFly (FlyID, FVendorID*, FName, FSalePrice, FPurchasePrice, FQty_OH, FQty_Commit, FReorder, FColorID*, FPatternID*, FSizeID*)

Foreign Key FVendorID references TVendor
Not Null
On Delete Restrict

Foreign Key FColorID references TColor
Not Null
On Delete Restrict

Foreign Key FPatternID references TPattern
Not Null
On Delete Restrict

Foreign Key FSizeID references TSize
Not Null
On Delete Restrict

TFlyKit (FKID, FKFlyID*, FKFlyID*, FKQuantity)

Foreign Key FKFlyID references TFly
Not Null
On Delete Restrict

Foreign Key FKFlyID references TFly
Not Null
On Delete Restrict

TRole (RoleID, Role)

TEmployee (EmpID, RoleID*, EFirstName, ELastName, EAddress, ECity, EStateID*, EPostal, EPhone, EEmail)

Foreign Key ERoleID references TRole
Not Null
On Delete Restrict

Foreign Key EStateID references TState

Not Null
On Delete Restrict

TDeliveryOut (DelID, DOPiEmpID*, DOPaEmpID*, DOShEmpID*, ShipDate)

Foreign Key DOPiEmpID references TEmployee
Not Null
On Delete Restrict

Foreign Key DOPaEmpID references TEmployee
Null Allowed
On Delete Restrict

Foreign Key DOShEmpID references TEmployee
Null Allowed
On Delete Restrict

TCustomerType (CustTypeID, CustomerType)

TCustomer (CustID, CCustTypeID*, CFirstName, CLastName, CAddress, CCity, CStateID*, CPostal, CPhone, CEmail, MailList)

Foreign Key CCustTypeID references TCustomerType
Not Null
On Delete Restrict

Foreign Key CStateID references TState
Not Null
On Delete Restrict

TSalesChannel (SCID, ChannelType)

TPaymentIn (PayInID, PIEmpID*, PIDate, PIAmount, PayType)

Foreign Key PIEmpID references TEmployee
Not Null
On Delete Restrict

TSalesOrder (SOID, CustID*, EmpID*, SCID*, PayInID*, SODate)

Foreign Key SOPayInID references TPaymentIn
Not Null
On Delete Restrict

Foreign Key SOCustID references TCustomer

Not Null
On Delete Restrict

Foreign Key SOEmpID references TEmployee
Null Allowed
On Delete Restrict

Foreign Key SOSCID references TSalesChannel
Not Null
On Delete Restrict

TDiscount (DiscountID, Discount)

TAppliedDiscount (ADID, ADSOID*, DiscountID*, ADTimesApplied, ADDate)

Foreign Key ADSOID references TSalesOrder
Null Allowed
On Delete Restrict

Foreign Key ADDiscountID references TDiscount
Not Null
On Delete Restrict

TSalesOrderLine (SOLID, SOLFlyID*, SOLSOID*, SOLDelID*, SOLQuantity, SOLDate)

Foreign Key SOLFlyID references TFly
Not Null
On Delete Restrict

Foreign Key SOLSOID references TSalesOrder
Not Null
On Delete Restrict

Foreign Key SOLDelID references TDeliveryOut
Null Allowed
On Delete Restrict

TTrip (TripID, TSOID*, TEmpID*, TDate, GroupSize)

Foreign Key TSOID references TSalesOrder
Null Allowed
On Delete Restrict

Foreign Key TEmpID references TEmployee
Not Null
On Delete Restrict

TPaymentOut (PayOutID, PayOEmpID*, Invoice#, PayOAmount, PayODate)

Foreign Key PayOEmpID references TEmployee
Not Null
On Delete Restrict

TPurchaseOrder (POID, POEmpID*, POPayOutID*, POVendID*, PODate)

Foreign Key POEmpID references TEmployee
Not Null
On Delete Restrict

Foreign Key POPayOutID references TPaymentOut
Null Allowed
On Delete Restrict

Foreign Key POVendID references TVendor
Not Null
On Delete Restrict

TPurchaseOrderLine (POLID, POLPOID*, POLFlyID*, POLDelID*, POLQuantity, POLDate)

Foreign Key POLPOID references TPurchaseOrder
Not Null
On Delete Restrict

Foreign Key POLFlyID references TFly
Not Null
On Delete Restrict

Differences between ERD and Normalized Relations

When it comes to ERDs and Normalized relations, it is important to understand how they serve a distinct role but are also interconnected in the overall design of a database. An ERD is a visual representation of entities within a system and the relationships, showing how the data is organized and linked. On the other hand, normalized relations involve the process of organizing the data in the database to reduce redundancies and dependencies, which ensures data integrity. Normalization entails breaking down the database until it is atomic, this eliminates redundancy and anomalies, like update, insertion, and deletion anomalies. While an ERD offers a high-level overview of the database structure, normalized relations dive deeper into optimizing data storage and retrieval by minimizing redundancies and ensuring data integrity.

Referential Integrity

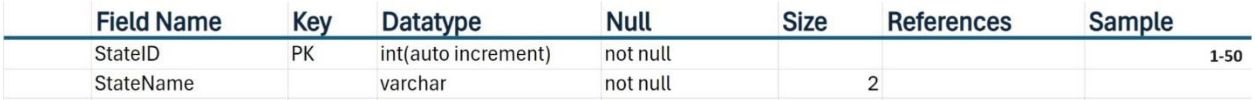
In a relational database, referential integrity constraint ensures the accuracy of data relationships between various tables. It is essential that every foreign key in one table must have a corresponding primary key value in another table, or it is null. This ensures that the relationships between tables are valid, and that the data is consistent throughout the database. The significance of referential integrity constraints is maintaining data integrity and preventing isolated records. By enforcing this rule, the database ensures that any changes or deletions to primary key values are accurately reflected in the tables and avoids data inconsistencies or any anomalies. Overall, referential integrity constraint helps maintain the integrity of relationships between tables, ensuring the accuracy and reliability of the database.

Physical Design and Implementation

After conceptual design and logical design comes physical design. Physical design is the actual creation of the database. It identifies the best way to integrate the design made by the previous steps into an RDBMS with the goal of creating an efficient database. This is the only party of the process that is platform specific. In this case, we created our database in SQL Server. Here, “relations” finally become “tables”, “columns” to “fields”, and “rows” to “records”. By using the normalized relations to make a data dictionary, the requirements and specifications needed for everything to be implemented are defined.

Data Dictionary

A data dictionary is an output of the physical design process, which reflects all the decisions made about how the database is to be implemented. This dictionary contains all table, table names, attributes, primary keys, foreign keys, data type, size, null or not null, references, sample size, and descriptions. It is important that all this information be translated into a format that can be implemented into an RDBMS. In the dictionary's construction, these must be reflected accurately for an effective dictionary. We need to ensure the accuracy of each table, its name, and its attributes. With the tables created we need to ensure that the keys, whether primary or foreign, are correctly stated in the dictionary. Alongside the keys of the tables, it is essential to state which tables reference which with foreign keys. Datatypes are needed for each attribute; this is significant because if we have the incorrect data type it will not reflect accurate records. There are many different data types, so it is important that we implement the correct ones for each attribute. In the dictionary, it is essential to know the size of each attribute to ensure there is enough data. Lastly, descriptions of each attribute are needed to see how the data will reflect in the database.



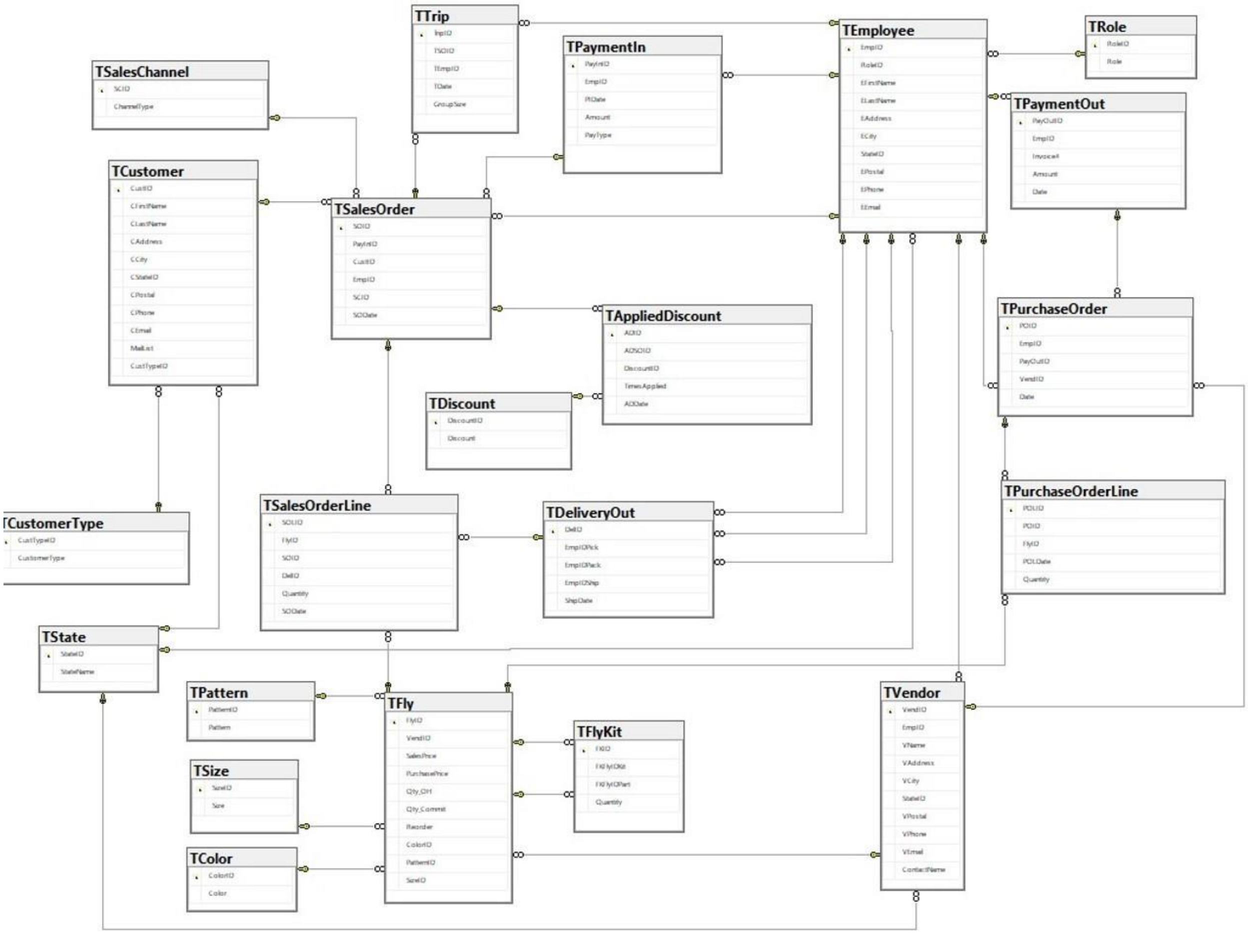
Field Name	Key	Datatype	Null	Size	References	Sample
StateID	PK	int(auto increment)	not null			1-50
StateName		varchar	not null	2		

As for examples in the data dictionary for Elysian Fly Fishing there are ample amounts of data. One of the easiest examples to look at from this project would be one of the reference tables. We can look at the State table shown above. The State table has two fields, a primary key (PK) and a state name. The PK is an integer, and the state name is stored as characters. Both these values cannot be null. There are 50 states which make for 50 primary keys, and a size of 2 for state name as they will be abbreviated.

Denormalization

Denormalization is taking a step backwards in the normalization process. Through the database's creation, we went up to the third normal form but there can be over a dozen more if taken further. Denormalization can be helpful in big databases where separated data and atomicity can slow down the database and make it less efficient. Normally we focus on breaking down data and making sure everything within a table relates to itself, and this can cause more tables to be made. This is typically for the best, but when dealing with a large database it can create long loading times when executing queries. Through our case, we did not have to use denormalization as the database size and requirements did not make it necessary.

Implemented Physical Design



UOKA0965

Challenges Faced/Addressed During Implementation

Our team put a lot of time into the implementation phase. We are all new to this subject, so we ran into some challenges. A big challenge we faced was interacting with the RDBMS platform. It was our first time using it and our team was having trouble seeing the same things and figuring out where everything was. This was a huge setback and a big learning curve we had to overcome. While not being easy, it was no match for our determination and patience. With some time, we got through it. Another challenge faced during implementation was the lack of actual visualization. The other steps of the process are easy to look at and read, but this is a bit lacking when it comes to the actual implementation. Once we create something, it is not just right in front of us as what we are used to. It is more difficult to see previous work, making it difficult to see how certain parts we reached. This was only made possible by teamwork and collaboration as we constantly asked each other for advice and questions.

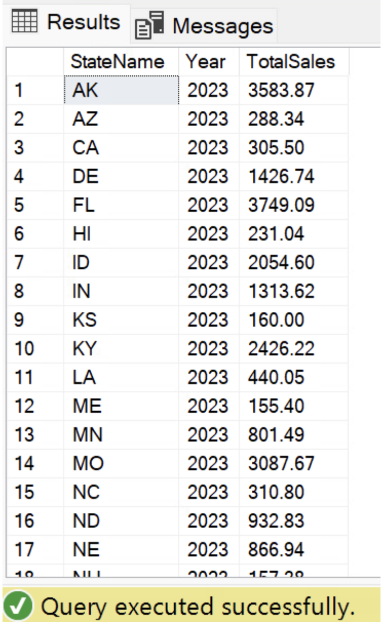
Strengths and Weaknesses Encountered During Implementation

Our team's strengths shined through this part of the process. At times we faced trouble, we never questioned the possibility but rather tackled it head on. Our team proved to be great at problem solving. When something did not make sense, we would bat around ideas until somebody figured it out. This would create a domino effect allowing us all to improve and become better at implementation. Something we excelled at was implementing actual data. This could be because we were more familiar with the table names. Writing queries was also a strength for us as this is something some of us are very comfortable doing and interpreting. One weakness we did face was the initialization of the database, creating the tables initially. This is the important framework that holds all the information. This is something we all had to slowly pick at and learn until we grew more familiar with it.

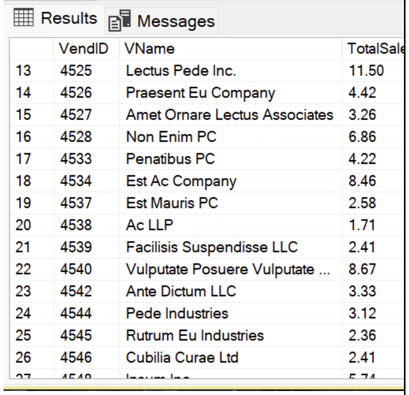
Specific SQL Statements Requested

<table>
<tr><th>Query #</th><th>Question</th><th>SQL</th><th>Partial Output</th></tr>
<tr><td>1</td><td>Total sales (in dollars) by customer state per year (e.g., total sales for all customers from Montana, Wyoming, Colorado, etc.).</td><td>SELECT StateName, Year(so.SODate) as Year, Sum(SalesPrice*Quantity) as TotalSales

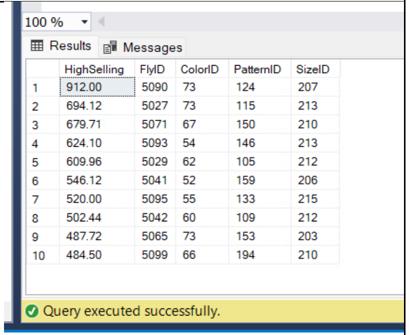
FROM TState s Join TCustomer c on s.StateID = c.CStateID Join TSalesOrder so on c.CustID = so.CustID Join TSalesOrderLine sol on so.SOID = sol.SOID Join TFly f on sol.FlyID = f.FlyID

GROUP BY StateName, Year(so.SODate)</td><td>
Results Messages
<table>
<tr><th></th><th>StateName</th><th>Year</th><th>TotalSales</th></tr>
<tr><td>1</td><td>AK</td><td>2023</td><td>3583.87</td></tr>
<tr><td>2</td><td>AZ</td><td>2023</td><td>288.34</td></tr>
<tr><td>3</td><td>CA</td><td>2023</td><td>305.50</td></tr>
<tr><td>4</td><td>DE</td><td>2023</td><td>1426.74</td></tr>
<tr><td>5</td><td>FL</td><td>2023</td><td>3749.09</td></tr>
<tr><td>6</td><td>HI</td><td>2023</td><td>231.04</td></tr>
<tr><td>7</td><td>ID</td><td>2023</td><td>2054.60</td></tr>
<tr><td>8</td><td>IN</td><td>2023</td><td>1313.62</td></tr>
<tr><td>9</td><td>KS</td><td>2023</td><td>160.00</td></tr>
<tr><td>10</td><td>KY</td><td>2023</td><td>2426.22</td></tr>
<tr><td>11</td><td>LA</td><td>2023</td><td>440.05</td></tr>
<tr><td>12</td><td>ME</td><td>2023</td><td>155.40</td></tr>
<tr><td>13</td><td>MN</td><td>2023</td><td>801.49</td></tr>
<tr><td>14</td><td>MO</td><td>2023</td><td>3087.67</td></tr>
<tr><td>15</td><td>NC</td><td>2023</td><td>310.80</td></tr>
<tr><td>16</td><td>ND</td><td>2023</td><td>932.83</td></tr>
<tr><td>17</td><td>NE</td><td>2023</td><td>866.94</td></tr>
</table>
✓ Query executed successfully.</td></tr>
<tr><td>2</td><td>Total sales (in dollars) by vendor per year. We must be able to calculate profit (sale price - purchase price)</td><td>SELECT v.VendID, VName, Sum(SalesPrice - PurchasePrice) as TotalSales

FROM TFly f JOIN TVendor v on f.VendID = v.VendID

GROUP BY v.VendID, VName</td><td>
Results Messages
<table>
<tr><th></th><th>VendID</th><th>VName</th><th>TotalSale</th></tr>
<tr><td>13</td><td>4525</td><td>Lectus Pede Inc.</td><td>11.50</td></tr>
<tr><td>14</td><td>4526</td><td>Praesent Eu Company</td><td>4.42</td></tr>
<tr><td>15</td><td>4527</td><td>Amet Ornare Lectus Associates</td><td>3.26</td></tr>
<tr><td>16</td><td>4528</td><td>Non Enim PC</td><td>6.86</td></tr>
<tr><td>17</td><td>4533</td><td>Penatibus PC</td><td>4.22</td></tr>
<tr><td>18</td><td>4534</td><td>Est Ac Company</td><td>8.46</td></tr>
<tr><td>19</td><td>4537</td><td>Est Mauris PC</td><td>2.58</td></tr>
<tr><td>20</td><td>4538</td><td>Ac LLP</td><td>1.71</td></tr>
<tr><td>21</td><td>4539</td><td>Facilisis Suspendisse LLC</td><td>2.41</td></tr>
<tr><td>22</td><td>4540</td><td>Vulputate Posuere Vulputate ...</td><td>8.67</td></tr>
<tr><td>23</td><td>4542</td><td>Ante Dictum LLC</td><td>3.33</td></tr>
<tr><td>24</td><td>4544</td><td>Pede Industries</td><td>3.12</td></tr>
<tr><td>25</td><td>4545</td><td>Rutrum Eu Industries</td><td>2.36</td></tr>
<tr><td>26</td><td>4546</td><td>Cubilia Curae Ltd</td><td>2.41</td></tr>
</table></td></tr>
<tr><td>3</td><td>The ten highest selling (in dollars) (a) patterns, (b) sizes, (c) pattern-size-color combinations each year.</td><td>Select Distinct Top 10 SUM (TFLy.SalesPrice*Quantity) as HighSelling,TFLy.FlyID,TFLy.ColorID,TFLy.PatternID,TFLy.SizeID)

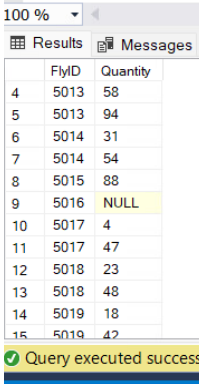
From TFly Inner Join TSalesOrderLIne as sol on TFLy.FlyID=sol.FlyID

Where (sol.SODate > ‘2023-12-31')</td><td>
100 %
Results Messages
<table>
<tr><th></th><th>HighSelling</th><th>FlyID</th><th>ColorID</th><th>PatternID</th><th>SizeID</th></tr>
<tr><td>1</td><td>912.00</td><td>5090</td><td>73</td><td>124</td><td>207</td></tr>
<tr><td>2</td><td>694.12</td><td>5027</td><td>73</td><td>115</td><td>213</td></tr>
<tr><td>3</td><td>679.71</td><td>5071</td><td>67</td><td>150</td><td>210</td></tr>
<tr><td>4</td><td>624.10</td><td>5093</td><td>54</td><td>146</td><td>213</td></tr>
<tr><td>5</td><td>609.96</td><td>5029</td><td>62</td><td>105</td><td>212</td></tr>
<tr><td>6</td><td>546.12</td><td>5041</td><td>52</td><td>159</td><td>206</td></tr>
<tr><td>7</td><td>520.00</td><td>5095</td><td>55</td><td>133</td><td>215</td></tr>
<tr><td>8</td><td>502.44</td><td>5042</td><td>60</td><td>109</td><td>212</td></tr>
<tr><td>9</td><td>487.72</td><td>5065</td><td>73</td><td>153</td><td>203</td></tr>
<tr><td>10</td><td>484.50</td><td>5099</td><td>66</td><td>194</td><td>210</td></tr>
</table>
✓ Query executed successfully.</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Group By TFly.FlyID,TFly.ColorID,TFLy.SIzeID,TFly.PatternID

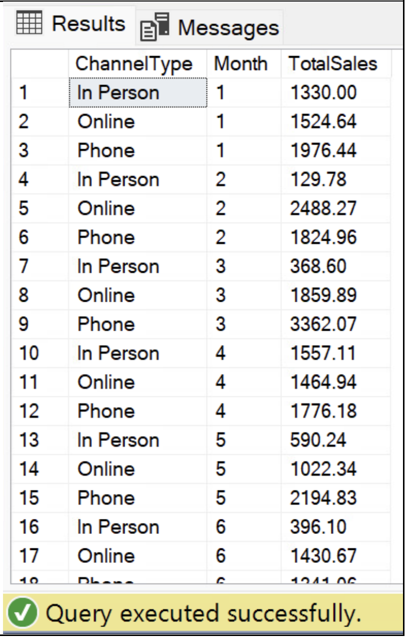
Order By HighSelling DESC</td><td></td></tr>
<tr><td>4</td><td>The number of times each product (fly) was sold. We want to see also those flies that have never been sold so that we can discontinue them</td><td>SELECT distinct TFLy.FlyID, SOL.Quantity,

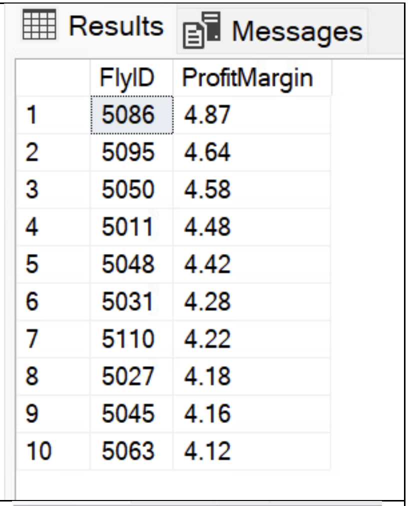
FROM TFLy LEFT JOIN TSalesOrderLine sol

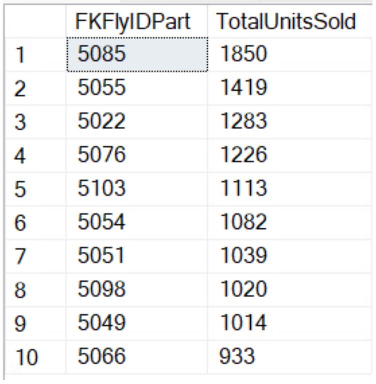
On TFLy.FlyID=sol.FlyID</td><td>
100 %
Results Messages
<table><tr><th></th><th>FlyID</th><th>Quantity</th></tr><tr><td>4</td><td>5013</td><td>58</td></tr><tr><td>5</td><td>5013</td><td>94</td></tr><tr><td>6</td><td>5014</td><td>31</td></tr><tr><td>7</td><td>5014</td><td>54</td></tr><tr><td>8</td><td>5015</td><td>88</td></tr><tr><td>9</td><td>5016</td><td>NULL</td></tr><tr><td>10</td><td>5017</td><td>4</td></tr><tr><td>11</td><td>5017</td><td>47</td></tr><tr><td>12</td><td>5018</td><td>23</td></tr><tr><td>13</td><td>5018</td><td>48</td></tr><tr><td>14</td><td>5019</td><td>18</td></tr><tr><td>15</td><td>5019</td><td>42</td></tr></table>
Query executed success</td></tr>
<tr><td>5</td><td>Total sales (in dollars) for each channel per month</td><td>SELECT DISTINCT ChannelType, Month(so.

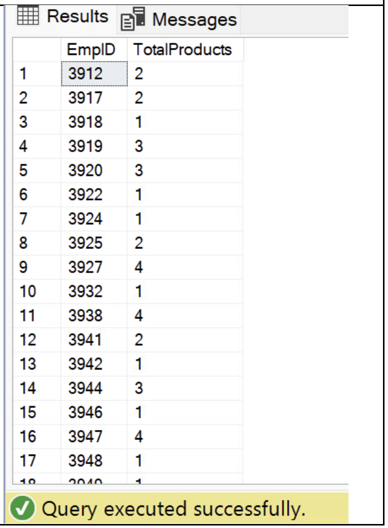
SODate) as Month, Sum(SalesPrice*Quantity) as TotalSales

FROM TSalesChannel sc JOIN TSalesOrder so on sc.SCID = so.SCID. JOIN TSalesOrderLine sol on so.SOID = sol.SOID JOIN TFly f on sol.FlyID = f.FlyID

GROUP BY ChannelType, Month(so.SODate)</td><td>
Results Messages
<table><tr><th></th><th>ChannelType</th><th>Month</th><th>TotalSales</th></tr><tr><td>1</td><td>In Person</td><td>1</td><td>1330.00</td></tr><tr><td>2</td><td>Online</td><td>1</td><td>1524.64</td></tr><tr><td>3</td><td>Phone</td><td>1</td><td>1976.44</td></tr><tr><td>4</td><td>In Person</td><td>2</td><td>129.78</td></tr><tr><td>5</td><td>Online</td><td>2</td><td>2488.27</td></tr><tr><td>6</td><td>Phone</td><td>2</td><td>1824.96</td></tr><tr><td>7</td><td>In Person</td><td>3</td><td>368.60</td></tr><tr><td>8</td><td>Online</td><td>3</td><td>1859.89</td></tr><tr><td>9</td><td>Phone</td><td>3</td><td>3362.07</td></tr><tr><td>10</td><td>In Person</td><td>4</td><td>1557.11</td></tr><tr><td>11</td><td>Online</td><td>4</td><td>1464.94</td></tr><tr><td>12</td><td>Phone</td><td>4</td><td>1776.18</td></tr><tr><td>13</td><td>In Person</td><td>5</td><td>590.24</td></tr><tr><td>14</td><td>Online</td><td>5</td><td>1022.34</td></tr><tr><td>15</td><td>Phone</td><td>5</td><td>2194.83</td></tr><tr><td>16</td><td>In Person</td><td>6</td><td>396.10</td></tr><tr><td>17</td><td>Online</td><td>6</td><td>1430.67</td></tr><tr><td>18</td><td>Phone</td><td>6</td><td>1341.06</td></tr></table>
Query executed successfully.</td></tr>
</table>

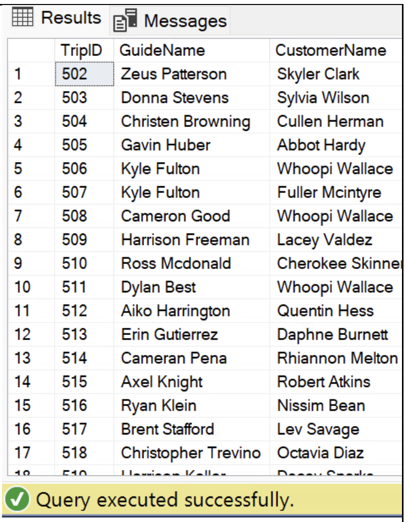
6	10% of the products that have the highest margin	SELECT TOP(10) PERCENT FlyID, MAX(SalesPrice-PurchasePrice) as ProfitMargin
		FROM TFly
		GROUP BY FlyID
		ORDER BY MAX(SalesPrice-PurchasePrice)
			
			FlyID ProfitMargin
			1 5086 4.87
			2 5095 4.64
			3 5050 4.58
			4 5011 4.48
			5 5048 4.42
			6 5031 4.28
			7 5110 4.22
			8 5027 4.18
			9 5045 4.16
			10 5063 4.12

7	The ten most popular (units sold) DIY fly-tying materials.	SELECT Top 10 fk.FKFlyIDPart, sum(sol.Quantity) as TotalUnitsSold
		FROM TSalesOrderLine sol JOIN TFly f on sol.FlyID = f.FlyID JOIN TFlyKit fk on fk.FKFlyIDKit = f.FlyID
		GROUP BY FKFlyIDPart
		ORDER BY TotalUnitsSold desc
			FKFlyIDPart TotalUnitsSold
			1 5085 1850
			2 5055 1419
			3 5022 1283
			4 5076 1226
			5 5103 1113
			6 5054 1082
			7 5051 1039
			8 5098 1020
			9 5049 1014
			10 5066 933

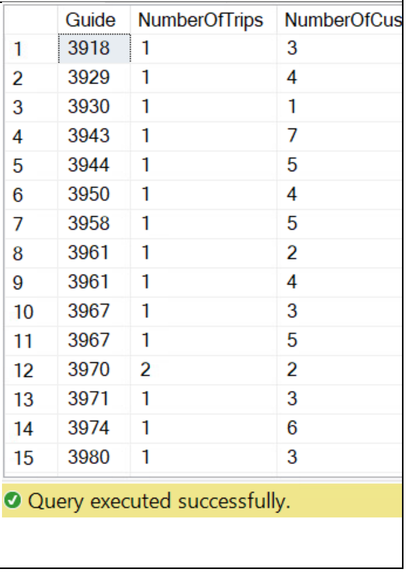
8	The number of distinct products managed by each vendor manager.	SELECT Distinct EmpID, Count(FlyID) as TotalProducts
		FROM TVendor v join TFly f on v.VendID = f.VendID
		GROUP BY EmpID
			
			EmpID TotalProducts
			1 3912 2
			2 3917 2
			3 3918 1
			4 3919 3
			5 3920 3
			6 3922 1
			7 3924 1
			8 3925 2
			9 3927 4
			10 3932 1
			11 3938 4
			12 3941 2
			13 3942 1
			14 3944 3
			15 3946 1
			16 3947 4
			17 3948 1
			18 3949 1
			Query executed successfully.

<table>
<tr>
<td>9</td>
<td>The upcoming, scheduled guided trips (i.e., the guided trips that have already been sold) for each guide, including the guide’s name, the trip destination, the customer name, and the number in the customer’s party.

*client request no destination</td>
<td>SELECT TripID, EFirstName + ‘ ‘ + ELastName as GuideName, CFirstName + ‘ ‘ + CLastName as CustomerName, GroupSize

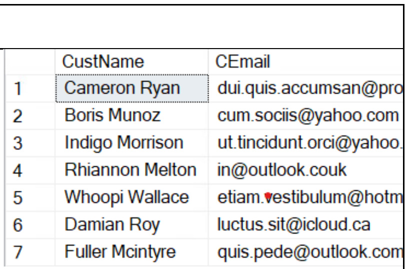
FROM TEmployee e JOIN TTrip t on e.EmpID = t.TEmpID JOIN TSalesOrder so on t.TSOID = so.SOID JOIN TCustomer c on so.CustID = c.CustID</td>
<td>
Results Messages
<table>
<tr><th></th><th>TripID</th><th>GuideName</th><th>CustomerName</th></tr>
<tr><td>1</td><td>502</td><td>Zeus Patterson</td><td>Skyler Clark</td></tr>
<tr><td>2</td><td>503</td><td>Donna Stevens</td><td>Sylvia Wilson</td></tr>
<tr><td>3</td><td>504</td><td>Christen Browning</td><td>Cullen Herman</td></tr>
<tr><td>4</td><td>505</td><td>Gavin Huber</td><td>Abbot Hardy</td></tr>
<tr><td>5</td><td>506</td><td>Kyle Fulton</td><td>Whoopi Wallace</td></tr>
<tr><td>6</td><td>507</td><td>Kyle Fulton</td><td>Fuller Mcintyre</td></tr>
<tr><td>7</td><td>508</td><td>Cameron Good</td><td>Whoopi Wallace</td></tr>
<tr><td>8</td><td>509</td><td>Harrison Freeman</td><td>Lacey Valdez</td></tr>
<tr><td>9</td><td>510</td><td>Ross Mcdonald</td><td>Cherokee Skinne</td></tr>
<tr><td>10</td><td>511</td><td>Dylan Best</td><td>Whoopi Wallace</td></tr>
<tr><td>11</td><td>512</td><td>Aiko Harrington</td><td>Quentin Hess</td></tr>
<tr><td>12</td><td>513</td><td>Erin Gutierrez</td><td>Daphne Burnett</td></tr>
<tr><td>13</td><td>514</td><td>Cameran Pena</td><td>Rhiannon Melton</td></tr>
<tr><td>14</td><td>515</td><td>Axel Knight</td><td>Robert Atkins</td></tr>
<tr><td>15</td><td>516</td><td>Ryan Klein</td><td>Nissim Bean</td></tr>
<tr><td>16</td><td>517</td><td>Brent Stafford</td><td>Lev Savage</td></tr>
<tr><td>17</td><td>518</td><td>Christopher Trevino</td><td>Octavia Diaz</td></tr>
</table>
✓ Query executed successfully.</td>
</tr>
<tr>
<td>10</td>
<td>Number of trips and the number of customers taken on fishing trips by each guide in the past 6 months</td>
<td>SELECT EmpID AS Guide, COUNT(T.TripID) AS NumberOfTrips, GroupSize AS NumberOfCustomers

FROM TTrip T JOIN TSalesOrder SO ON T.SOID = SO.SOID JOIN TCustomer C ON SO.CustID = C.CustID

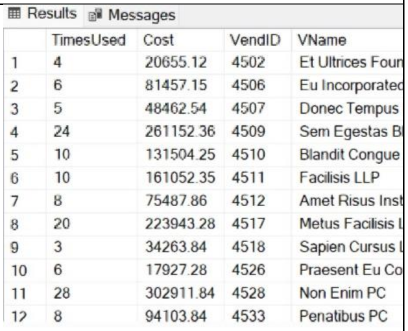
WHERE T.TDate BETWEEN ’11-30-2023’ AND ’04-30-2024’</td>
<td>
<table>
<tr><th></th><th>Guide</th><th>NumberOfTrips</th><th>NumberOfCus</th></tr>
<tr><td>1</td><td>3918</td><td>1</td><td>3</td></tr>
<tr><td>2</td><td>3929</td><td>1</td><td>4</td></tr>
<tr><td>3</td><td>3930</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td>3943</td><td>1</td><td>7</td></tr>
<tr><td>5</td><td>3944</td><td>1</td><td>5</td></tr>
<tr><td>6</td><td>3950</td><td>1</td><td>4</td></tr>
<tr><td>7</td><td>3958</td><td>1</td><td>5</td></tr>
<tr><td>8</td><td>3961</td><td>1</td><td>2</td></tr>
<tr><td>9</td><td>3961</td><td>1</td><td>4</td></tr>
<tr><td>10</td><td>3967</td><td>1</td><td>3</td></tr>
<tr><td>11</td><td>3967</td><td>1</td><td>5</td></tr>
<tr><td>12</td><td>3970</td><td>2</td><td>2</td></tr>
<tr><td>13</td><td>3971</td><td>1</td><td>3</td></tr>
<tr><td>14</td><td>3974</td><td>1</td><td>6</td></tr>
<tr><td>15</td><td>3980</td><td>1</td><td>3</td></tr>
</table>
✓ Query executed successfully.</td>
</tr>
</table>

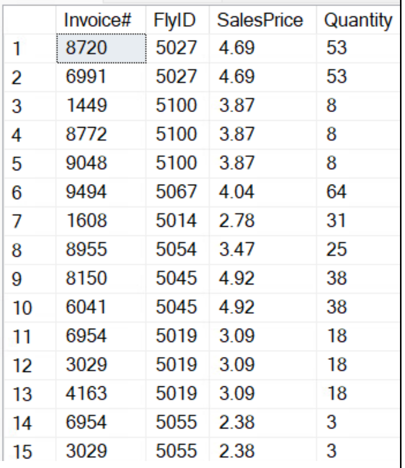
<table>
<tr><td></td><td></td><td>GROUP BY GroupSize, EmpID</td><td></td></tr>
<tr><td>11</td><td>Names and email addresses of all customers who made purchases in a given month. We need to be able to enter the month</td><td>SELECT concat(CFirstname, ‘ ‘, CLastname) as CustName, CEmail

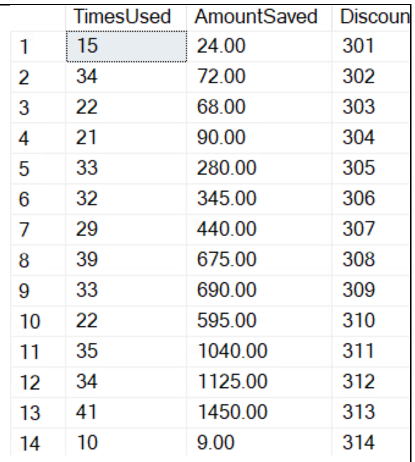
FROM TCustomer c JOIN TSalesOrder so on c.CustID = so.CustID

WHERE month(SODate) = ‘10’</td><td><table><tr><th></th><th>CustName</th><th>CEmail</th></tr><tr><td>1</td><td>Cameron Ryan</td><td>dui.quis.accumsan@pro</td></tr><tr><td>2</td><td>Boris Munoz</td><td>cum.sociis@yahoo.com</td></tr><tr><td>3</td><td>Indigo Morrison</td><td>ut.tincidunt.orci@yahoo.</td></tr><tr><td>4</td><td>Rhiannon Melton</td><td>in@outlook.couk</td></tr><tr><td>5</td><td>Whoopi Wallace</td><td>etiam.vestibulum@hotm</td></tr><tr><td>6</td><td>Damian Roy</td><td>luctus.sit@icloud.ca</td></tr><tr><td>7</td><td>Fuller Mcintyre</td><td>quis.pede@outlook.com</td></tr></table></td></tr>
<tr><td>12</td><td>Number of times used and dollars spent on each shipping vendor and shipping type by vendor</td><td>SELECT Count(PO.VendID) TimesUsed, Sum(POL.Quantity*F.PurchasePrice) Cost, PO.VendID, V.VName

FROM TVendor V JOIN TPurchaseOrder PO ON PO.VendID=V.VendID JOIN TFly F ON F.VendID=V.VendID JOIN TPurchaseOrderLine POL ON POL.POID=PO.POID

GROUP BY PO.VendID, V.VName</td><td> <table><tr><th></th><th>TimesUsed</th><th>Cost</th><th>VendID</th><th>VName</th></tr><tr><td>1</td><td>4</td><td>20655.12</td><td>4502</td><td>Et Ultrices Foun</td></tr><tr><td>2</td><td>6</td><td>81457.15</td><td>4506</td><td>Eu Incorporated</td></tr><tr><td>3</td><td>5</td><td>48462.54</td><td>4507</td><td>Donec Tempus</td></tr><tr><td>4</td><td>24</td><td>261152.36</td><td>4509</td><td>Sem Egestas Bl</td></tr><tr><td>5</td><td>10</td><td>131504.25</td><td>4510</td><td>Blandit Congue</td></tr><tr><td>6</td><td>10</td><td>161052.35</td><td>4511</td><td>Facilisis LLP</td></tr><tr><td>7</td><td>8</td><td>75487.86</td><td>4512</td><td>Amet Risus Inst</td></tr><tr><td>8</td><td>20</td><td>223943.28</td><td>4517</td><td>Metus Facilisis L</td></tr><tr><td>9</td><td>3</td><td>34263.84</td><td>4518</td><td>Sapien Cursus L</td></tr><tr><td>10</td><td>6</td><td>17927.28</td><td>4526</td><td>Praesent Eu Co</td></tr><tr><td>11</td><td>28</td><td>302911.84</td><td>4528</td><td>Non Enim PC</td></tr><tr><td>12</td><td>8</td><td>94103.84</td><td>4533</td><td>Penatibus PC</td></tr></table></td></tr>
<tr><td>13</td><td>Invoice lines for a given sales invoice number and given customer name</td><td>SELECT [Invoice#], f.FlyID, f.SalesPrice, sol.Quantity, c.CustID

FROM TCustomer c JOIN TSalesOrder so on c.CustID = so.CustID JOIN TSalesOrderLine sol on so.SOID = sol.SOID JOIN TFly f on sol.FlyID = f.FlyID JOIN TVendor v on v.VendID = f.VendID JOIN TPurchaseOrder po on po.VendID = v.VendID JOIN TPaymentOut payo on payo.PayOutID = po.PayOutID</td><td><table><tr><th></th><th>Invoice#</th><th>FlyID</th><th>SalesPrice</th><th>Quantity</th></tr><tr><td>1</td><td>8720</td><td>5027</td><td>4.69</td><td>53</td></tr><tr><td>2</td><td>6991</td><td>5027</td><td>4.69</td><td>53</td></tr><tr><td>3</td><td>1449</td><td>5100</td><td>3.87</td><td>8</td></tr><tr><td>4</td><td>8772</td><td>5100</td><td>3.87</td><td>8</td></tr><tr><td>5</td><td>9048</td><td>5100</td><td>3.87</td><td>8</td></tr><tr><td>6</td><td>9494</td><td>5067</td><td>4.04</td><td>64</td></tr><tr><td>7</td><td>1608</td><td>5014</td><td>2.78</td><td>31</td></tr><tr><td>8</td><td>8955</td><td>5054</td><td>3.47</td><td>25</td></tr><tr><td>9</td><td>8150</td><td>5045</td><td>4.92</td><td>38</td></tr><tr><td>10</td><td>6041</td><td>5045</td><td>4.92</td><td>38</td></tr><tr><td>11</td><td>6954</td><td>5019</td><td>3.09</td><td>18</td></tr><tr><td>12</td><td>3029</td><td>5019</td><td>3.09</td><td>18</td></tr><tr><td>13</td><td>4163</td><td>5019</td><td>3.09</td><td>18</td></tr><tr><td>14</td><td>6954</td><td>5055</td><td>2.38</td><td>3</td></tr><tr><td>15</td><td>3029</td><td>5055</td><td>2.38</td><td>3</td></tr></table></td></tr>
</table>

14	A number of times a discount was applied to a sales order. List all the information about the discount, the total amount saved by customers that used the discount	SELECT SUM(AD.TimesApplied) TimesUsed, SUM(Discount) AmountSaved, D.DiscountID
		FROM TSalesOrder SO JOIN TAppliedDiscount AD ON SO.SOID=AD.ADSOID JOIN TDiscount D ON D.DiscountID=AD.DiscountID
		ORDER BY AmountSaved
			
			TimesUsed AmountSaved Discoun
			1 15 24.00 301
			2 34 72.00 302
			3 22 68.00 303
			4 21 90.00 304
			5 33 280.00 305
			6 32 345.00 306
			7 29 440.00 307
			8 39 675.00 308
			9 33 690.00 309
			10 22 595.00 310
			11 35 1040.00 311
			12 34 1125.00 312
			13 41 1450.00 313
			14 10 9.00 314

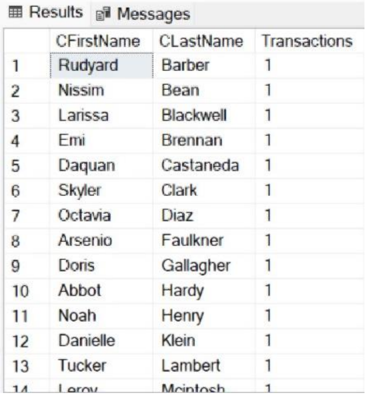
Three Additional Queries

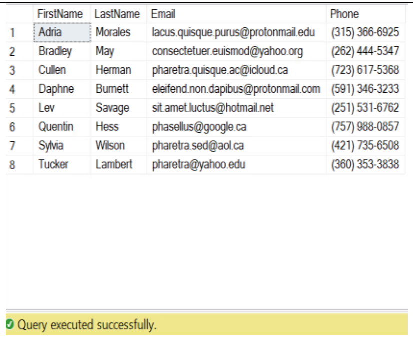
With the requested 14 queries above, we went ahead and added an additional 3. We decided to do this to show additional information that could be found by interacting with this database. We are very pleased with what we have created and do not want any feature to be overlooked. We want the client to feel like the database is more than sufficient and to be able to query anything imagined.

<table>
<tr><th>Query #</th><th>Question</th><th>Importance/Findings</th><th>SQL</th><th>Partial Output</th></tr>
<tr><td>1</td><td>Which customers are on the mailing list and have only made one purchase?</td><td>Mail can be sent to encourage one-time customers to revisit. Several customers were found.</td><td>SELECT C.CFirstName, C.CLastName, COUNT(SO.CustID) Transactions

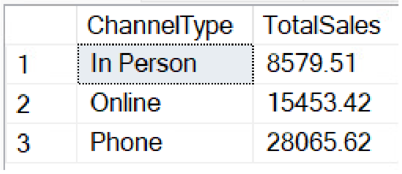
FROM TCustomer C JOIN TSalesOrder SO ON C.CustID=SO.CustID WHERE C.MailList=’Yes’

GROUP BY C.CFirstName, C.CLastName

HAVING COUNT(SO.CustID)=1</td><td>
Results Messages
<table>
<tr><th></th><th>CFirstName</th><th>CLastName</th><th>Transactions</th></tr>
<tr><td>1</td><td>Rudyard</td><td>Barber</td><td>1</td></tr>
<tr><td>2</td><td>Nissim</td><td>Bean</td><td>1</td></tr>
<tr><td>3</td><td>Larissa</td><td>Blackwell</td><td>1</td></tr>
<tr><td>4</td><td>Emi</td><td>Brennan</td><td>1</td></tr>
<tr><td>5</td><td>Daquan</td><td>Castaneda</td><td>1</td></tr>
<tr><td>6</td><td>Skyler</td><td>Clark</td><td>1</td></tr>
<tr><td>7</td><td>Octavia</td><td>Diaz</td><td>1</td></tr>
<tr><td>8</td><td>Arsenio</td><td>Faulkner</td><td>1</td></tr>
<tr><td>9</td><td>Doris</td><td>Gallagher</td><td>1</td></tr>
<tr><td>10</td><td>Abbot</td><td>Hardy</td><td>1</td></tr>
<tr><td>11</td><td>Noah</td><td>Henry</td><td>1</td></tr>
<tr><td>12</td><td>Danielle</td><td>Klein</td><td>1</td></tr>
<tr><td>13</td><td>Tucker</td><td>Lambert</td><td>1</td></tr>
<tr><td>14</td><td>Leroy</td><td>Mcintosh</td><td>1</td></tr>
</table></td></tr>
<tr><td>2</td><td>Names, email addresses, and phone numbers of customers who purchased fly kits in the month of January. Order by customer name</td><td>Allows a view of a list of customers and their contact information that purchased a specific item in a given month</td><td>SELECT Distinct CFirstName AS FirstName, CLastName AS LastName, CEmail AS Email, CPhone AS Phone

FROM TCustomer C JOIN TSalesOrder SO ON C.CustID = SO.CustID JOIN TSalesOrderLine SOL ON SO.SOID = SOL.SOID JOIN TFly F ON SOL.FlyID = F.FlyID</td><td>
<table>
<tr><th></th><th>FirstName</th><th>LastName</th><th>Email</th><th>Phone</th></tr>
<tr><td>1</td><td>Adria</td><td>Morales</td><td>lacus.quisque.purus@protonmail.edu</td><td>(315) 366-6925</td></tr>
<tr><td>2</td><td>Bradley</td><td>May</td><td>consectetuer.euismod@yahoo.org</td><td>(262) 444-5347</td></tr>
<tr><td>3</td><td>Cullen</td><td>Herman</td><td>pharetra.quisque.ac@icloud.ca</td><td>(723) 617-5368</td></tr>
<tr><td>4</td><td>Daphne</td><td>Burnett</td><td>eleifend.non.dapibus@protonmail.com</td><td>(591) 346-3233</td></tr>
<tr><td>5</td><td>Lev</td><td>Savage</td><td>sit.amet.luctus@hotmail.net</td><td>(251) 531-6762</td></tr>
<tr><td>6</td><td>Quentin</td><td>Hess</td><td>phasellus@google.ca</td><td>(757) 988-0857</td></tr>
<tr><td>7</td><td>Sylvia</td><td>Wilson</td><td>pharetra.sed@aol.ca</td><td>(421) 735-6508</td></tr>
<tr><td>8</td><td>Tucker</td><td>Lambert</td><td>pharetra@yahoo.edu</td><td>(360) 353-3838</td></tr>
</table>
Query executed successfully.</td></tr>
</table>

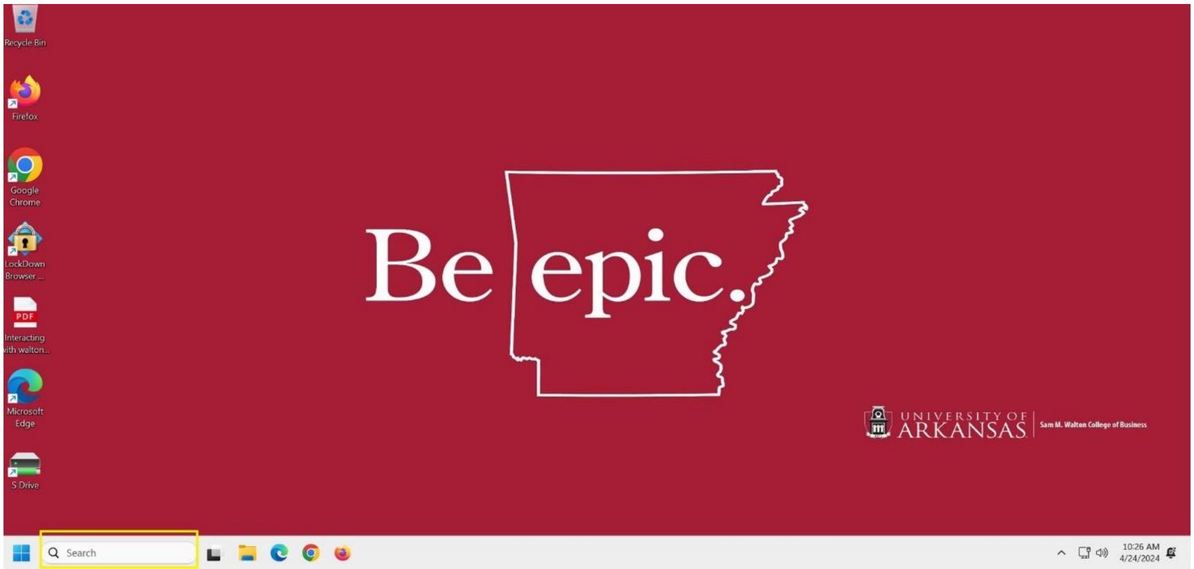
			JOIN TFlyKit FK ON FK.FKFlyIDKit - F.FlyID
			WHERE month(SO.SODate) = ‘1’
			ORDER BY CFirstName, CLastName	

3	Which Sales Channel has the highest profit?	Allows us to see which Sales Channel is the most popular by looking at the sum of profits	SELECT distinct ChannelType, SUM(SalesPrice*Quantity) AS TotalSales
			FROM TSalesChannel sh JOIN TSalesOrder so ON sh.SCID = so.SCID JOIN TSalesOrderLine sol ON so.SOID = sol.SOID JOIN TFly f ON sol.FlyID = f.FlyID
			GROUP BY ChannelType
				
				ChannelType TotalSales
				1 In Person 8579.51
				2 Online 15453.42
				3 Phone 28065.62

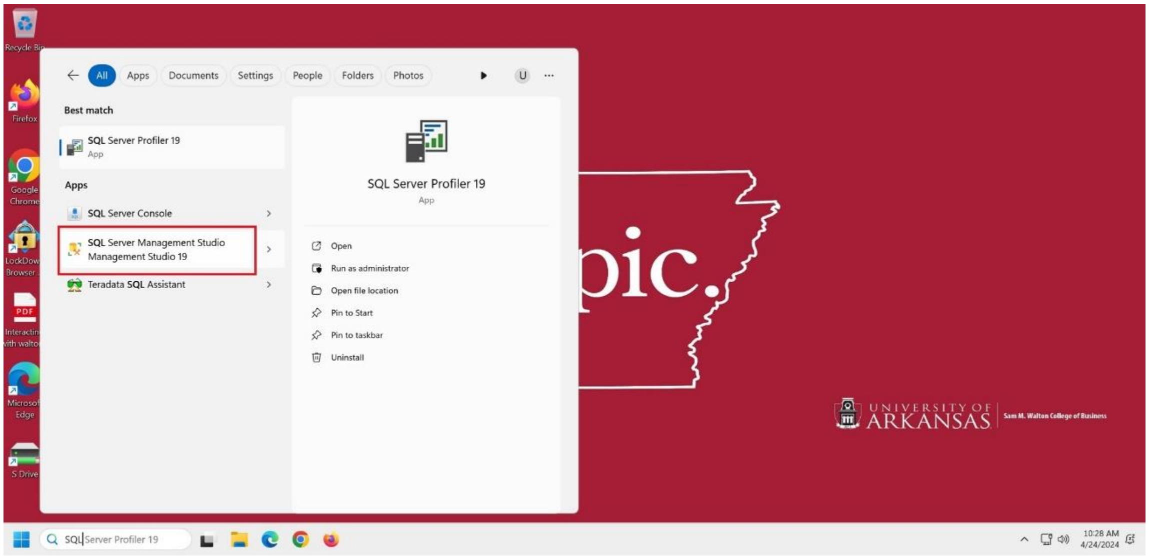
User Documentation

Interacting with a database may not be easy, especially for the first time. We created a step-by-step guide that shows how to reach and interact with the database from first turning on the desktop. Our team prioritizes the transition of the database from our hands to the client's.

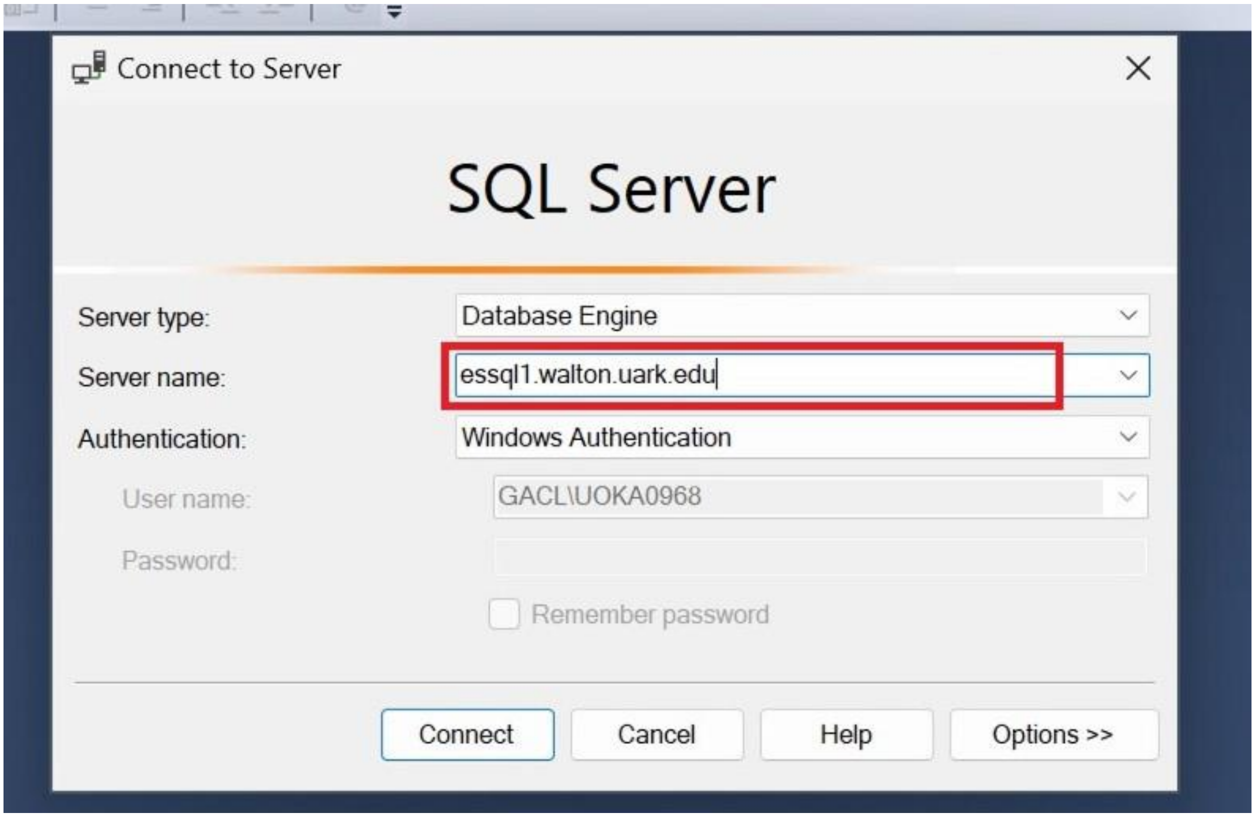
Step 1: Navigate to the windows search bar in the bottom left corner



Step 2: Search and Double click SQL Server Management Studio 19



Step 3: After SQL Server loads, the window below will pop up. In server name, type "essql1.walton.uark.edu" and press connect



Connect to Server

SQL Server

Server type: Database Engine

Server name: essql1.walton.uark.edu

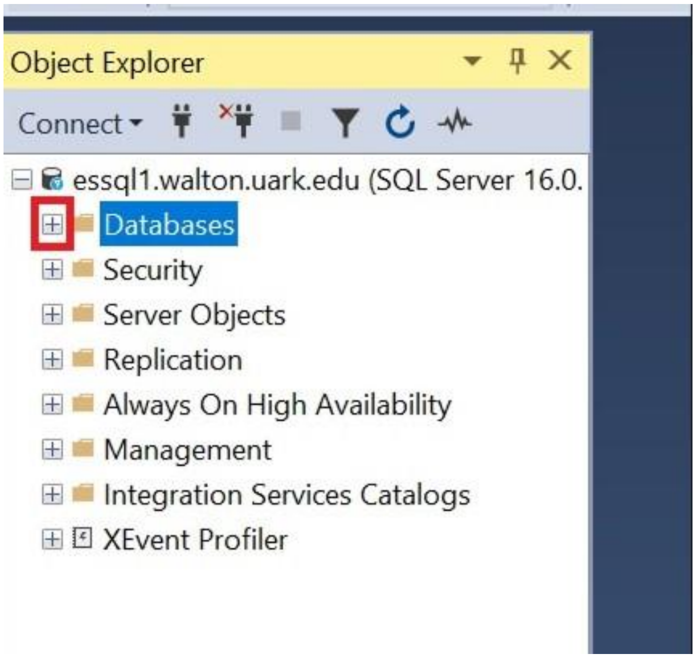
Authentication: Windows Authentication

User name: GACL\UOKA0968

Password:

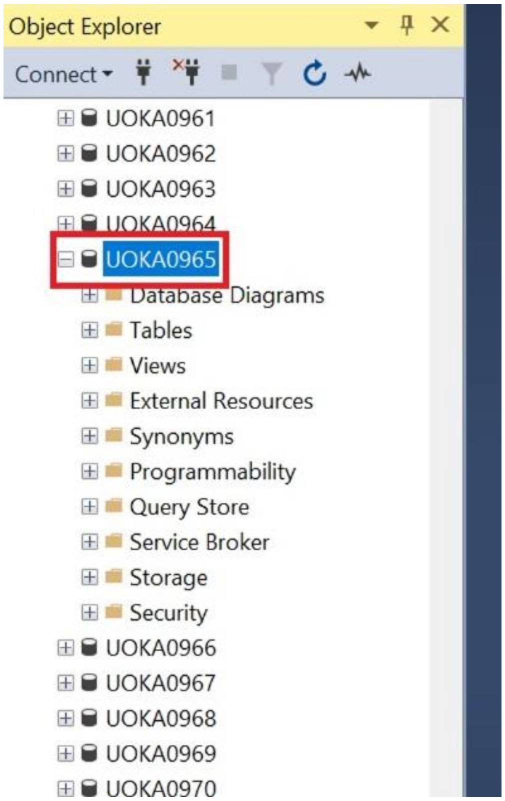
□ Remember password

Step 4: On the left should be this panel. Click the expand icon next to databases

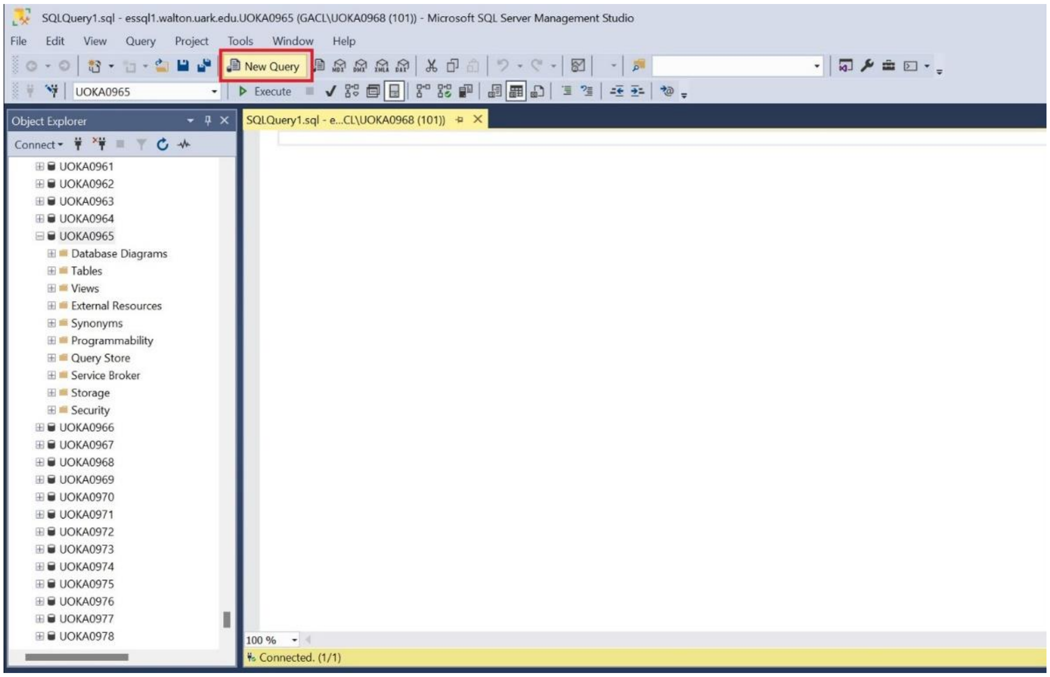


Object Explorer
Connect
essql1.walton.uark.edu (SQL Server 16.0.
Databases
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
XEvent Profiler

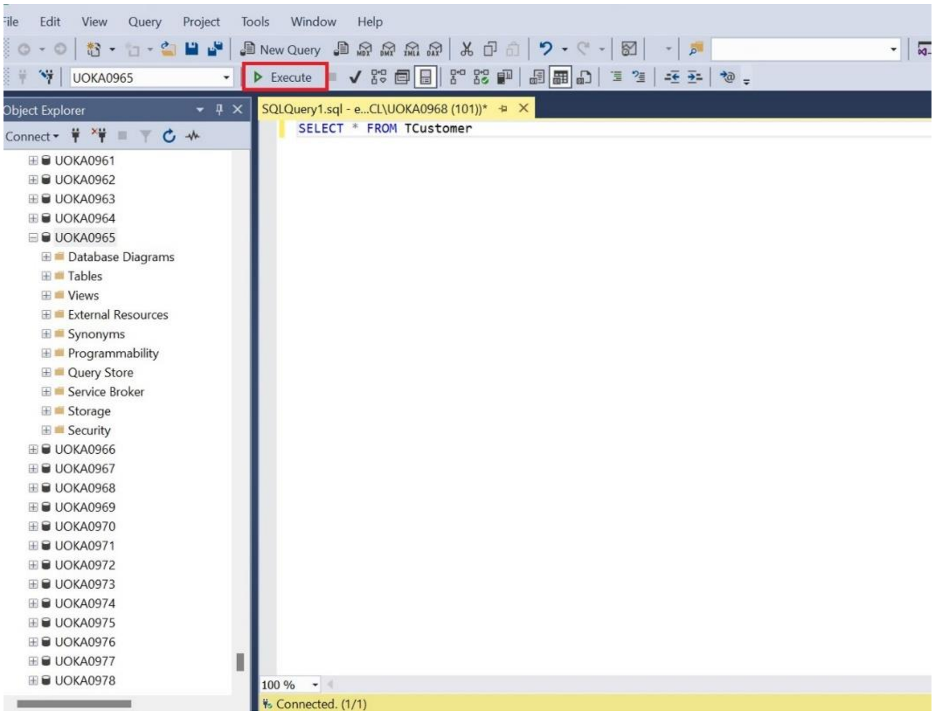
Step 5: Scroll or search to your database “UOKA0965”



Step 6: You are now in your database! To run a query, press “New Query” at the top which will open a blank page for you to work on



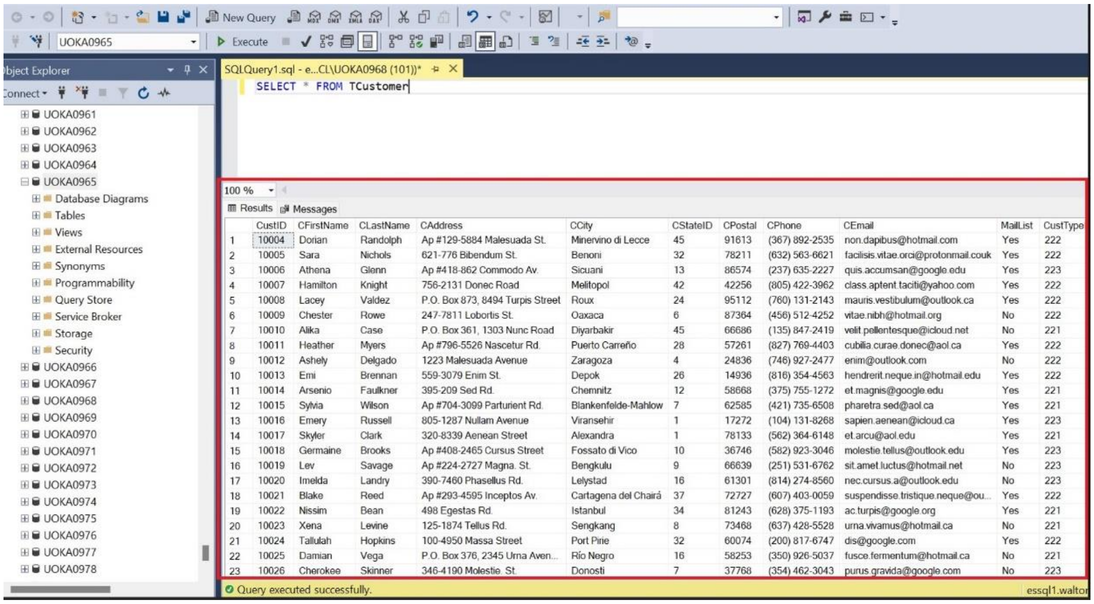
Step 7: Type your query in the white space and press “Execute” when finished to run your query. Below is an example to view all data in the Customer Table



```
SELECT * FROM TCustomer
```

Connected. (1/1)

Step 8: After executing, a panel at the bottom of the screen will pop up with information pertaining to your query. With the example shown below, we see the entire Customer table. The top row contains the field names with their respective records below. On the left we get a number attribute listing the number of records.



```
SELECT * FROM TCustomer
```

	CustID	CFirstName	CLastName	CAddress	CCity	CStateID	CPostal	CPhone	CEmail	MailList	CustType
1	10004	Dorian	Randolph	Ap #129-5884 Malesuada St.	Minervino di Lecce	45	91613	(367) 892-2535	non.dapibus@hotmail.com	Yes	222
2	10005	Sara	Nichols	621-776 Bibendum St.	Benoni	32	78211	(632) 563-6621	facilisis.vitae.orci@protonmail.couk	Yes	222
3	10006	Athena	Glenn	Ap #418-862 Commodo Av.	Sicuani	13	86574	(237) 635-2227	quis.accumsan@google.edu	Yes	223
4	10007	Hamilton	Knight	756-2131 Donec Road	Melitopol	42	42256	(805) 422-3962	class.aptent.taciti@yahoo.com	Yes	222
5	10008	Lacey	Valdez	P.O. Box 873, 8494 Turpis Street	Roux	24	95112	(760) 131-2143	mauris.vestibulum@outlook.ca	Yes	222
6	10009	Chester	Rowe	247-7811 Lobortis St.	Oaxaca	6	87364	(456) 512-4252	vitae.nibh@hotmail.org	No	222
7	10010	Alika	Case	P.O. Box 361, 1303 Nunc Road	Diyarbakir	45	66686	(135) 847-2419	velit.pellentesque@icloud.net	No	221
8	10011	Heather	Myers	Ap #796-5526 Nascetur Rd.	Puerto Carreño	28	57261	(827) 769-4403	cubilia.curae.donec@aol.ca	Yes	222
9	10012	Ashely	Delgado	1223 Malesuada Avenue	Zaragoza	4	24836	(746) 927-2477	enim@outlook.com	No	222
10	10013	Emi	Brennan	559-3079 Enim St.	Depok	26	14936	(816) 354-4563	hendrerit.neque.in@hotmail.edu	Yes	222
11	10014	Arsenio	Faulkner	395-209 Sed Rd.	Chemnitz	12	58668	(375) 755-1272	et.magnis@google.edu	Yes	221
12	10015	Sylvia	Wilson	Ap #704-3099 Parturient Rd.	Blankenfelde-Mahlow	7	62585	(421) 735-6508	pharetra.sed@aol.ca	Yes	221
13	10016	Emery	Russell	805-1287 Nullam Avenue	Viransehir	1	17272	(104) 131-8268	sapien.aenean@icloud.ca	Yes	223
14	10017	Skyler	Clark	320-8339 Aenean Street	Alexandra	1	78133	(562) 364-6148	et.arcu@aol.edu	Yes	221
15	10018	Germaine	Brooks	Ap #408-2465 Cursus Street	Fossato di Vico	10	36746	(582) 923-3046	molestie.tellus@outlook.edu	Yes	223
16	10019	Lev	Savage	Ap #224-2727 Magna. St.	Bengkulu	9	66639	(251) 531-6762	sit.amet.luctus@hotmail.net	No	223
17	10020	Imelda	Landry	390-7460 Phasellus Rd.	Lelystad	16	61301	(814) 274-8560	nec.cursus.a@outlook.edu	No	223
18	10021	Blake	Reed	Ap #293-4595 Inceptos Av.	Cartagena del Chairá	37	72727	(607) 403-0059	suspendisse.tristique.neque@ou...	Yes	222
19	10022	Nissim	Bean	498 Egestas Rd.	Istanbul	34	81243	(628) 375-1193	ac.turpis@google.org	Yes	221
20	10023	Xena	Levine	125-1874 Tellus Rd.	Sengkang	8	73468	(637) 428-5528	urna.vivamus@hotmail.ca	No	221
21	10024	Tallulah	Hopkins	100-4950 Massa Street	Port Pirie	32	60074	(200) 817-6747	dis@google.com	Yes	222
22	10025	Damian	Vega	P.O. Box 376, 2345 Urna Aven...	Río Negro	16	58253	(350) 926-5037	fusce.fermentum@hotmail.ca	No	221
23	10026	Cherokee	Skinner	346-4190 Molestie. St.	Donosti	7	37768	(354) 462-3043	purus.gravida@google.com	No	223

Query executed successfully.

What We Learned Throughout This Process

This process was a great learning experience for our team. It was our first time working with a group to make a database. At times it was not easy because we were working with each other and with an RDBSM. This took some getting used to but eventually we learned how to efficiently navigate it with multiple hands. We learned how to split up the workload for this long process and how to stay on track. It is easy to get lost during the process so working as a group was essential to getting through it.

Bryston Stark	Throughout my experience with this project, I learned a lot. It was beneficial being able to use what I have learned with actual software and code. Before this process I felt like I had only known about the material and subject. After I feel as if I know how this material can be used and how it is applied. I was familiar with how to type and use SQL, but I never actually knew how to use an actual RDBSM. This process familiarized me with SQL Server, and I feel much more comfortable navigating it than I ever have. It is a very complex system that requires lots of collaboration when working with a team because it is easy to get behind. The error codes helped greatly to find problems and to troubleshoot.
Sara Hernandez	With all the content that was discussed in this course I feel like I have obtained all the concepts! The two sections that I really enjoyed were conceptual design and physical design. I truly enjoyed the physical design portion of the course and enjoyed working in SQL Server. Conceptual design was also a fun section for me. Getting to read and interpret the case and then creating an ERD in LucidChart was very interesting. There was a lot of build up until physical design and implementation, despite this each step of the way was clear and easy to follow. Working in SQL Server was by far my favorite part, getting to create tables and importing data was quite enjoyable. Overall, my experience with this project was enjoyable and I most definitely will use all this information in my future.
Blake Garrett	The information retained throughout this experience has been very interesting as well as advantageous. Prior to this course, I was unaware of the complexity and detail that went behind creating databases. I can now say with confidence that I feel comfortable with the material that we have covered. The structure of this course, starting with SQL later leading into full implementation into a database was very beneficial and will certainly be useful moving forward. I enjoyed learning how to use Lucid Chart for conceptual design with the creation of ERDs but SQL Server was what I found the most satisfaction in. There were difficult moments when trying to find tedious errors in the system. However, working to solve these complications within the team and tying everything together in order to understand what makes the server function properly was very rewarding. I am very pleased with what our project has resulted in and I am excited moving forward with these new skills.

Reed Brackett	The experience of creating a database for Elysian has not only been enjoyable, but something I can utilize to further my career. With Big Data becoming even more prevalent throughout major corporations, knowing the steps to create a database and applying it with SQL Server is a major boost to my skillset. This experience will open more opportunities for me in my professional career. At first, the idea of this project was extremely daunting, but the project has been arranged to be completed while learning the material. Solving the complex problems of the database and teaching myself how to use the tools in a more advanced way has been extremely rewarding.
Coreyonn Stokes	Throughout this project, I learned a good amount from making and learning from mistakes. My understanding of SQL's and ERD formatting had been fairly amateur prior to this project as I only ever worked with no more than the 5 or 6 entities so the room for error was very small. However, when working on this project I learned that with the more data you have the room for error also increases which is where I learned to be tedious in everything that I do especially when writing the codes in the SQL server. The process of working with a team has also expanded my learning as everyone specializes in certain areas, so whatever skillsets I may lack or get behind on, my team fills that gap and I pick up from them.I always understood the concept of coding from prior learnings, but I never knew how it could be used in a real working environment, which is what this project revealed to me. I'm optimistic that the skills/information I gained from this will transfer over and be used when it is time for me to get into my career.

Appendix

Team Contract

Team Motto: Do not be a stranger.

Team Members

Name	Email	Phone	Strengths	Availability to Meet
Bryston Stark	Bryston.a.stark-1@ou.edu	(832)916-5899	Communication, respect, determined	MW After 6pm T, TH After 3pm F All day
Sara Hernandez	Sarahernandez31@ou.edu	(469) 500-9388	Effective communication and problem solving	MW after class T, TH after 1:30pm F after 12:30
Coreyonn Stokes	coreyonn.c.stokes.-1@ou.edu	214-552-2683	Problem Solving Open Minded Communication	MW after class F anytime T TH varies
Thomas "Reed" Brackett	Thomas.r.brackett-1@ou.edu	469-263-8357	Communication, critical thinking, teachable	MW after 6 pm T, TH after 3 pm F all day
Blake Garrett	Blake.b.garrett-1@ou.edu	(405) 464-7065	Approachability, Dependability, Efficiency	T, TH anytime before 4:30

Team Expectations for the confidential peer evaluation:

Effort, effective communication, brainstorming, problem solving

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

Consistently not showing up to meetings, going no contact, laziness, not contributing

Data Dictionary Model

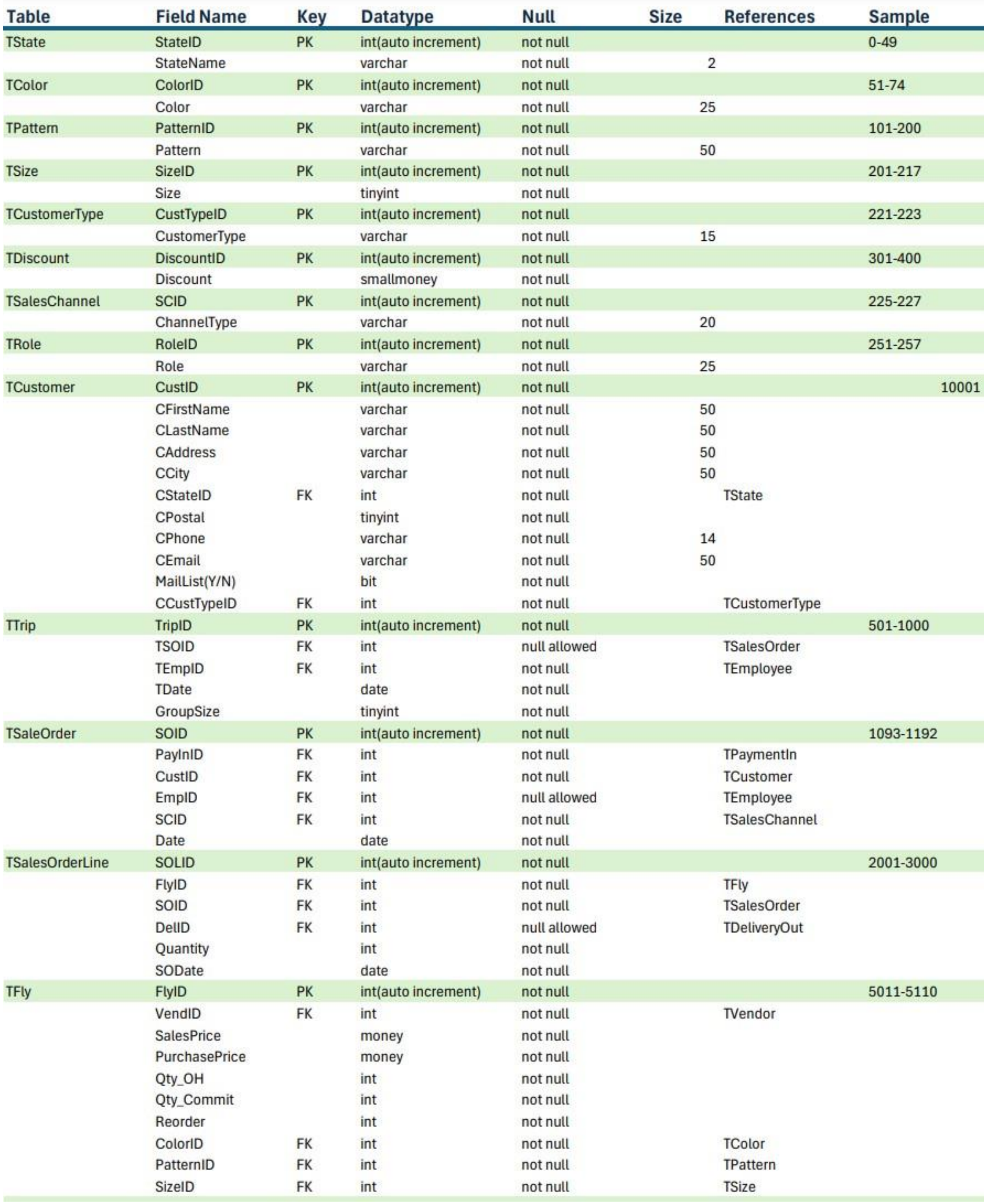
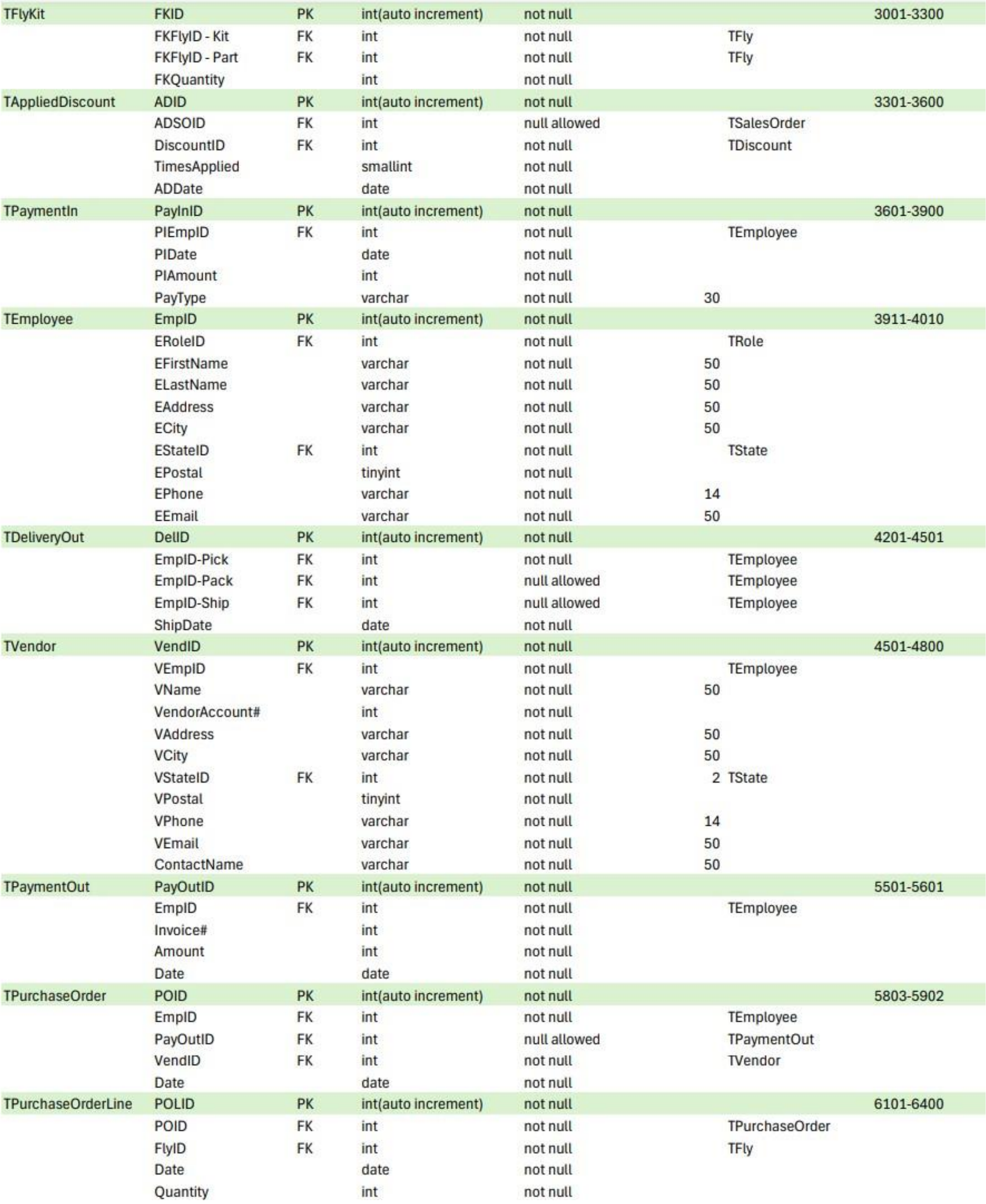
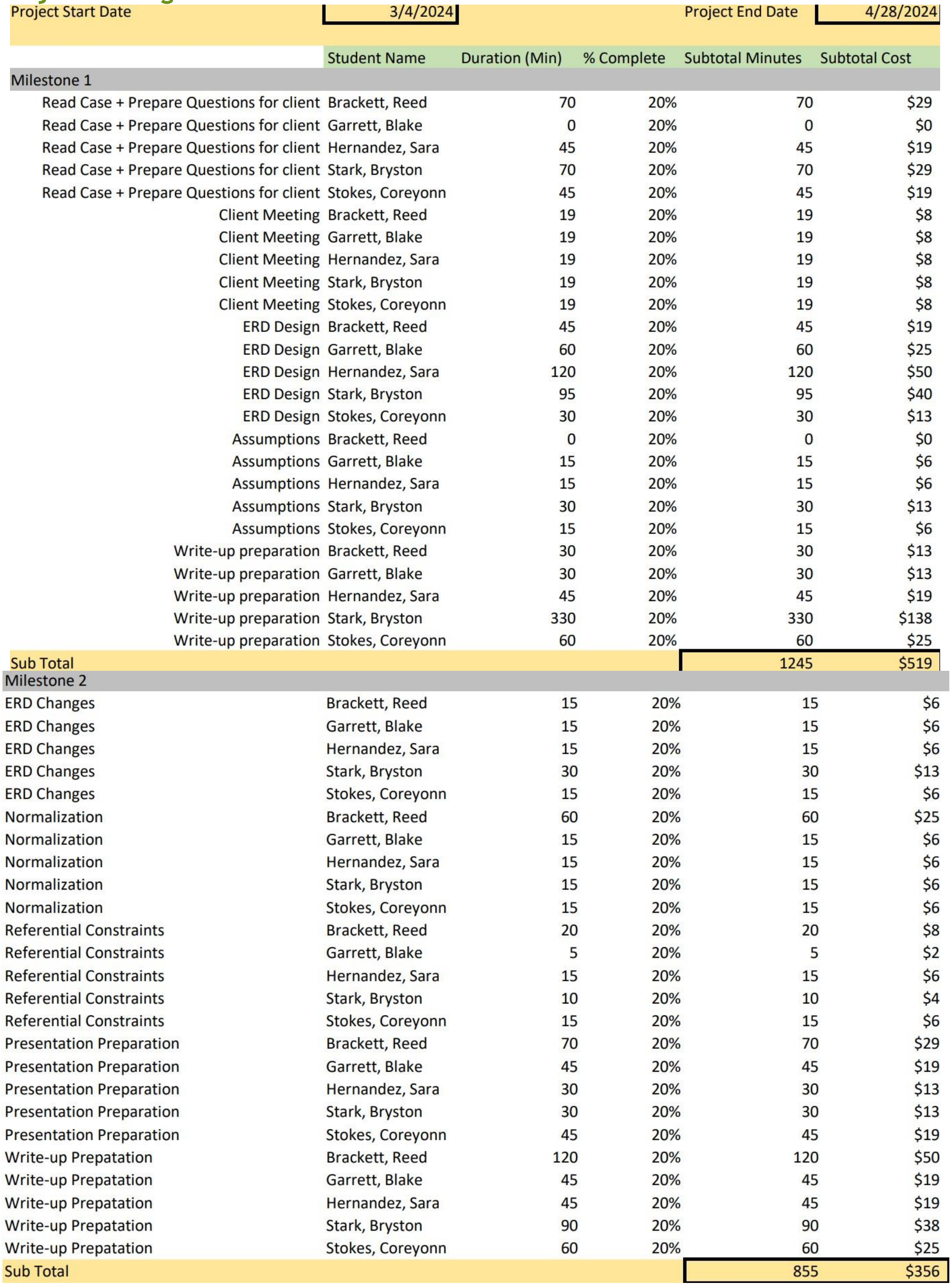


Table	Field Name	Key	Datatype	Null	Size	References	Sample
TState	StateID	PK	int(auto increment)	not null			0-49
	StateName		varchar	not null	2		
TColor	ColorID	PK	int(auto increment)	not null			51-74
	Color		varchar	not null	25		
TPattern	PatternID	PK	int(auto increment)	not null			101-200
	Pattern		varchar	not null	50		
TSize	SizeID	PK	int(auto increment)	not null			201-217
	Size		tinyint	not null			
TCustomerType	CustTypeID	PK	int(auto increment)	not null			221-223
	CustomerType		varchar	not null	15		
TDiscount	DiscountID	PK	int(auto increment)	not null			301-400
	Discount		smallmoney	not null			
TSalesChannel	SCID	PK	int(auto increment)	not null			225-227
	ChannelType		varchar	not null	20		
TRole	RoleID	PK	int(auto increment)	not null			251-257
	Role		varchar	not null	25		
TCustomer	CustID	PK	int(auto increment)	not null			10001
	CFirstName		varchar	not null	50		
	CLastName		varchar	not null	50		
	CAddress		varchar	not null	50		
	CCity		varchar	not null	50		
	CStateID	FK	int	not null		TState	
	CPostal		tinyint	not null			
	CPhone		varchar	not null	14		
	CEmail		varchar	not null	50		
	MailList(Y/N)		bit	not null			
	CCustTypeID	FK	int	not null		TCustomerType	
TTrip	TripID	PK	int(auto increment)	not null			501-1000
	TSOID	FK	int	null allowed		TSalesOrder	
	TEmpID	FK	int	not null		TEmployee	
	TDate		date	not null			
	GroupSize		tinyint	not null			
TSaleOrder	SOID	PK	int(auto increment)	not null			1093-1192
	PayInID	FK	int	not null		TPaymentIn	
	CustID	FK	int	not null		TCustomer	
	EmpID	FK	int	null allowed		TEmployee	
	SCID	FK	int	not null		TSalesChannel	
	Date		date	not null			
TSalesOrderLine	SOLID	PK	int(auto increment)	not null			2001-3000
	FlyID	FK	int	not null		TFly	
	SOID	FK	int	not null		TSalesOrder	
	DelID	FK	int	null allowed		TDeliveryOut	
	Quantity		int	not null			
	SODate		date	not null			
TFly	FlyID	PK	int(auto increment)	not null			5011-5110
	VendID	FK	int	not null		TVendor	
	SalesPrice		money	not null			
	PurchasePrice		money	not null			
	Qty_OH		int	not null			
	Qty_Commit		int	not null			
	Reorder		int	not null			
	ColorID	FK	int	not null		TColor	
	PatternID	FK	int	not null		TPattern	
	SizeID	FK	int	not null		TSize	

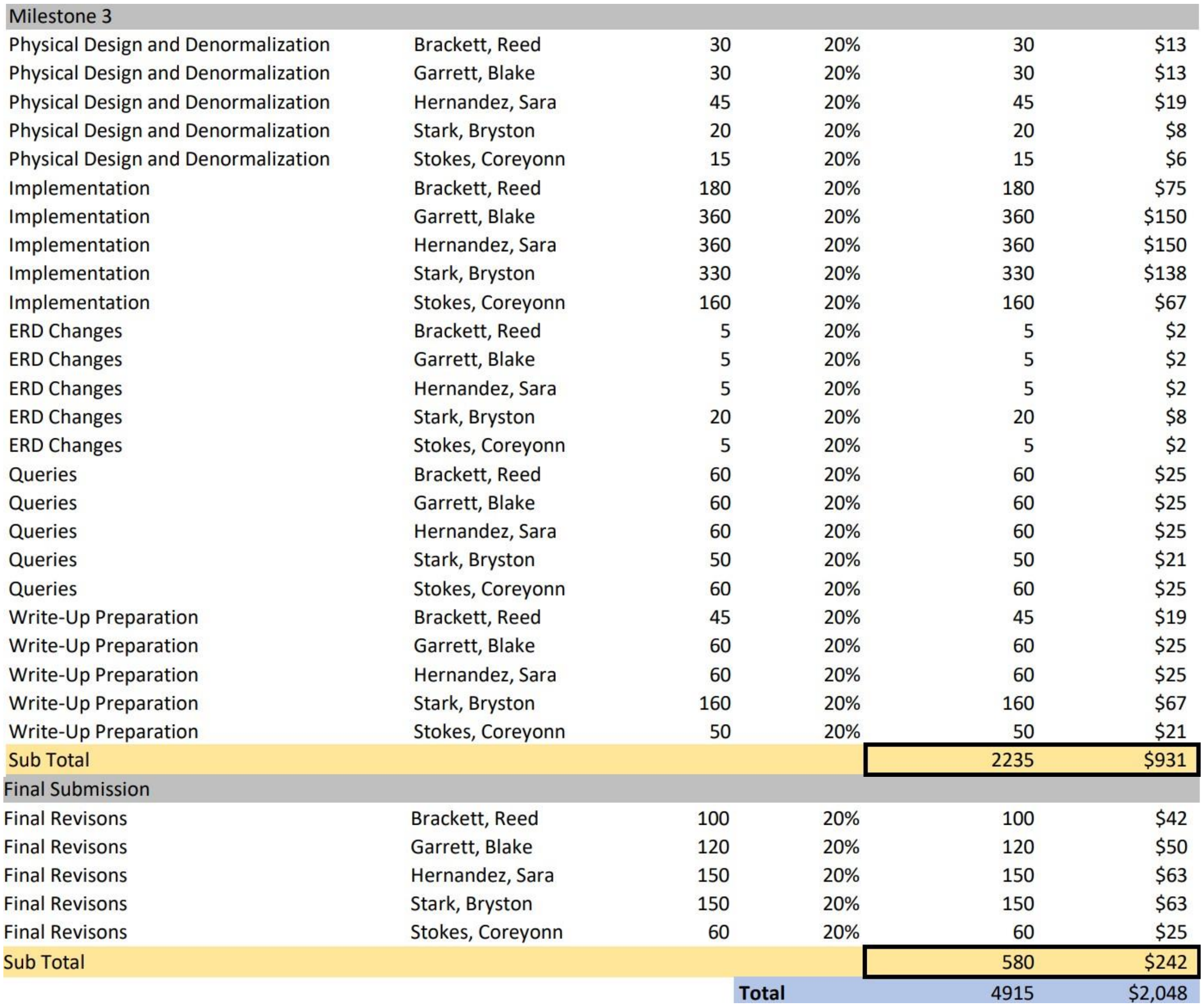


TFlyKit	FKID	PK	int(auto increment)	not null			3001-3300
	FKFlyID - Kit	FK	int	not null		TFly	
	FKFlyID - Part	FK	int	not null		TFly	
	FKQuantity		int	not null			
TAppliedDiscount	ADID	PK	int(auto increment)	not null			3301-3600
	ADSOID	FK	int	null allowed		TSalesOrder	
	DiscountID	FK	int	not null		TDiscount	
	TimesApplied		smallint	not null			
	ADDate		date	not null			
TPaymentIn	PayInID	PK	int(auto increment)	not null			3601-3900
	PIEmpID	FK	int	not null		TEmployee	
	PIDate		date	not null			
	PIAmount		int	not null			
	PayType		varchar	not null	30		
TEmployee	EmpID	PK	int(auto increment)	not null			3911-4010
	ERoleID	FK	int	not null		TRole	
	EFirstName		varchar	not null	50		
	ELastName		varchar	not null	50		
	EAddress		varchar	not null	50		
	ECity		varchar	not null	50		
	EStateID	FK	int	not null		TState	
	EPostal		tinyint	not null			
	EPhone		varchar	not null	14		
	EEmail		varchar	not null	50		
TDeliveryOut	DelID	PK	int(auto increment)	not null			4201-4501
	EmpID-Pick	FK	int	not null		TEmployee	
	EmpID-Pack	FK	int	null allowed		TEmployee	
	EmpID-Ship	FK	int	null allowed		TEmployee	
	ShipDate		date	not null			
TVendor	VendID	PK	int(auto increment)	not null			4501-4800
	VEmpID	FK	int	not null		TEmployee	
	VName		varchar	not null	50		
	VendorAccount#		int	not null			
	VAddress		varchar	not null	50		
	VCity		varchar	not null	50		
	VStateID	FK	int	not null	2	TState	
	VPostal		tinyint	not null			
	VPhone		varchar	not null	14		
	VEmail		varchar	not null	50		
	ContactName		varchar	not null	50		
TPaymentOut	PayOutID	PK	int(auto increment)	not null			5501-5601
	EmpID	FK	int	not null		TEmployee	
	Invoice#		int	not null			
	Amount		int	not null			
	Date		date	not null			
TPurchaseOrder	POID	PK	int(auto increment)	not null			5803-5902
	EmpID	FK	int	not null		TEmployee	
	PayOutID	FK	int	null allowed		TPaymentOut	
	VendID	FK	int	not null		TVendor	
	Date		date	not null			
TPurchaseOrderLine	POLID	PK	int(auto increment)	not null			6101-6400
	POID	FK	int	not null		TPurchaseOrder	
	FlyID	FK	int	not null		TFly	
	Date		date	not null			
	Quantity		int	not null			

Project Management



Project Start Date	3/4/2024			Project End Date	4/28/2024
	Student Name	Duration (Min)	% Complete	Subtotal Minutes	Subtotal Cost
Milestone 1					
Read Case + Prepare Questions for client	Brackett, Reed	70	20%	70	$29
Read Case + Prepare Questions for client	Garrett, Blake	0	20%	0	$0
Read Case + Prepare Questions for client	Hernandez, Sara	45	20%	45	$19
Read Case + Prepare Questions for client	Stark, Bryston	70	20%	70	$29
Read Case + Prepare Questions for client	Stokes, Coreyonn	45	20%	45	$19
Client Meeting	Brackett, Reed	19	20%	19	$8
Client Meeting	Garrett, Blake	19	20%	19	$8
Client Meeting	Hernandez, Sara	19	20%	19	$8
Client Meeting	Stark, Bryston	19	20%	19	$8
Client Meeting	Stokes, Coreyonn	19	20%	19	$8
ERD Design	Brackett, Reed	45	20%	45	$19
ERD Design	Garrett, Blake	60	20%	60	$25
ERD Design	Hernandez, Sara	120	20%	120	$50
ERD Design	Stark, Bryston	95	20%	95	$40
ERD Design	Stokes, Coreyonn	30	20%	30	$13
Assumptions	Brackett, Reed	0	20%	0	$0
Assumptions	Garrett, Blake	15	20%	15	$6
Assumptions	Hernandez, Sara	15	20%	15	$6
Assumptions	Stark, Bryston	30	20%	30	$13
Assumptions	Stokes, Coreyonn	15	20%	15	$6
Write-up preparation	Brackett, Reed	30	20%	30	$13
Write-up preparation	Garrett, Blake	30	20%	30	$13
Write-up preparation	Hernandez, Sara	45	20%	45	$19
Write-up preparation	Stark, Bryston	330	20%	330	$138
Write-up preparation	Stokes, Coreyonn	60	20%	60	$25
Sub Total				1245	$519
Milestone 2					
ERD Changes	Brackett, Reed	15	20%	15	$6
ERD Changes	Garrett, Blake	15	20%	15	$6
ERD Changes	Hernandez, Sara	15	20%	15	$6
ERD Changes	Stark, Bryston	30	20%	30	$13
ERD Changes	Stokes, Coreyonn	15	20%	15	$6
Normalization	Brackett, Reed	60	20%	60	$25
Normalization	Garrett, Blake	15	20%	15	$6
Normalization	Hernandez, Sara	15	20%	15	$6
Normalization	Stark, Bryston	15	20%	15	$6
Normalization	Stokes, Coreyonn	15	20%	15	$6
Referential Constraints	Brackett, Reed	20	20%	20	$8
Referential Constraints	Garrett, Blake	5	20%	5	$2
Referential Constraints	Hernandez, Sara	15	20%	15	$6
Referential Constraints	Stark, Bryston	10	20%	10	$4
Referential Constraints	Stokes, Coreyonn	15	20%	15	$6
Presentation Preparation	Brackett, Reed	70	20%	70	$29
Presentation Preparation	Garrett, Blake	45	20%	45	$19
Presentation Preparation	Hernandez, Sara	30	20%	30	$13
Presentation Preparation	Stark, Bryston	30	20%	30	$13
Presentation Preparation	Stokes, Coreyonn	45	20%	45	$19
Write-up Prepatation	Brackett, Reed	120	20%	120	$50
Write-up Prepatation	Garrett, Blake	45	20%	45	$19
Write-up Prepatation	Hernandez, Sara	45	20%	45	$19
Write-up Prepatation	Stark, Bryston	90	20%	90	$38
Write-up Prepatation	Stokes, Coreyonn	60	20%	60	$25
Sub Total				855	$356



Milestone 3					
Physical Design and Denormalization	Brackett, Reed	30	20%	30	$13
Physical Design and Denormalization	Garrett, Blake	30	20%	30	$13
Physical Design and Denormalization	Hernandez, Sara	45	20%	45	$19
Physical Design and Denormalization	Stark, Bryston	20	20%	20	$8
Physical Design and Denormalization	Stokes, Coreyonn	15	20%	15	$6
Implementation	Brackett, Reed	180	20%	180	$75
Implementation	Garrett, Blake	360	20%	360	$150
Implementation	Hernandez, Sara	360	20%	360	$150
Implementation	Stark, Bryston	330	20%	330	$138
Implementation	Stokes, Coreyonn	160	20%	160	$67
ERD Changes	Brackett, Reed	5	20%	5	$2
ERD Changes	Garrett, Blake	5	20%	5	$2
ERD Changes	Hernandez, Sara	5	20%	5	$2
ERD Changes	Stark, Bryston	20	20%	20	$8
ERD Changes	Stokes, Coreyonn	5	20%	5	$2
Queries	Brackett, Reed	60	20%	60	$25
Queries	Garrett, Blake	60	20%	60	$25
Queries	Hernandez, Sara	60	20%	60	$25
Queries	Stark, Bryston	50	20%	50	$21
Queries	Stokes, Coreyonn	60	20%	60	$25
Write-Up Preparation	Brackett, Reed	45	20%	45	$19
Write-Up Preparation	Garrett, Blake	60	20%	60	$25
Write-Up Preparation	Hernandez, Sara	60	20%	60	$25
Write-Up Preparation	Stark, Bryston	160	20%	160	$67
Write-Up Preparation	Stokes, Coreyonn	50	20%	50	$21
Sub Total				2235	$931
Final Submission					
Final Revisons	Brackett, Reed	100	20%	100	$42
Final Revisons	Garrett, Blake	120	20%	120	$50
Final Revisons	Hernandez, Sara	150	20%	150	$63
Final Revisons	Stark, Bryston	150	20%	150	$63
Final Revisons	Stokes, Coreyonn	60	20%	60	$25
Sub Total				580	$242
		Total		4915	$2,048

*$25/hour