Vince's Vinyl Database Management System

Tyler J. Latshaw

Southern New Hampshire University

Table of Contents

Vince's Vinyl Database Management System

For years, Vince Roberts has run a successful vinyl record business in the local university district without the need for any fancy technology, much like the records he sold. Despite the considerable depth of his inventory, Vince can recall all of the records he owned purely from his memory and the occasional notebook at his shop's counter. However, with how large his inventory has grown and the number of sales he receives in a day now, Vince has decided that it would be best to invest in a database system to support his business functions (Conger, 2014).

Organizational Analysis

Based on the information provided in the case study, it is readily apparent that Vince tends to run his business operations in a more traditional, old-school manner based on his lack of technology. Transitioning his entire business into a digital one with a robust database behind it will likely not be easy for him; however, it is an essential step to ensuring the future success of his day-to-day business. Before designing anything for the database or selecting a platform to run it, the database designer will need to closely analyze the entire organization, including its current challenges and how much they impact each department.

Current Organizational Challenges

Like any new technology package or solution, when developing a new database system, it is crucial to understand what the customer wants fully and what all the current challenges are, what business requirements need to be met, and what the limitations are the current system. By gathering this information and these requirements upfront, a complete database model can be designed and implemented correctly from the start instead of needing to go back and revise it multiple times.

Organization Problems

At its core, every challenge that Vince is having with his store can be traced back to not having a database in place to track vital information. Specifically, Vince does not have a properly established way to track inventory information, customer information, purchases made by customers, purchases made by Vince, and stock information. Currently, all of this information is either just in Vince's memory, documented by hand in a notebook, or not documented at all (Conger, 2014).

Tracking inventory and customer information is essential to implement in the database as it is the backbone of the entire business. At a minimum, according to the case text, inventory information would include the album's title and any applicable notes about it, like the condition or sales information such as its popularity (Conger, 2014). This information will later be used for Vince to look up a record if a customer is looking for a specific title to purchase. Regarding customer information, the case shows that Vince is documenting the customer's name and phone number only for when he is buying the record from a customer to resell, and only the customer's name for when he sells a record (Conger, 2014).

Moving forward, it is recommended and assumed that Vince will also collect information related to the records such as the artist and year and customer information like addresses and email addresses. This information would lead to a much more robust system and customer relationship model if he were also to implement a frontend interface or website integration in the future. Customers would be able to look at the current inventory in real-time or submit requests for albums without needing to go into the store.

Additional problems include not properly tracking both purchases from customers to resell and transactions where a customer is buying from him. In this case, it is noted that Vince

does not have a way to adequately know how much profit he is making on an item (Conger, 2014). By tracking these transactions, he would ensure he is always making money and even determine his current margins. He could even use this information to calculate pricing to ensure enough markup dynamically. Additionally, racking transactions would allow him to report his revenue for tax purposes adequately.

Business Requirements

The business requirements are fairly typical for a retail business from an operational standpoint. According to one technical business analyst and Certified Scrum Professional ScrumMaster (CSP-SM), defining business requirements upfront at the beginning of a new project can be one of the most important things to do when laying the groundwork for a project (W. Edwards, personal communication, October 22, 2021). The business requirements describe what the new database system will need to do, how it will support current operations, and who it will affect. Once the database system is implemented, it must satisfy all requirements to be accepted by the stakeholder and considered successful.

The main requirement for Vince's store is that the new database will need to accurately track all of the records that he has in his inventory with relevant information like the album title, artist, year, and any notes. He will also track information like the condition, pricing, and type of record as he sells different kinds and sizes of vinyl. Further, he will have a way to track key customer information such as names, phone numbers, addresses, and potentially more like if they are registered with him. The customer information will later be linked to track requests from the customers if they are looking for specific titles.

Lastly, it is required for Vince to be able to track all of his sales and purchases both from customers buying from him and he himself purchasing from customers to resell in his store

(Conger, 2014). From a technical standpoint, the database will need to maintain both integrity and security. This means that all of the data is secured from external influencers and remains accurate internally. Data should never be overwritten unless intentionally modified by a user or approved process, and all references and foreign keys between tables will remain correct using referential integrity properties (Oracle, n.d.). If the data is not accurate, the entire system will not be beneficial for Vince as it provides invalid reporting data that could negatively impact his business.

Current System Limitations

Identifying the limitations of Vince's current system is relatively straightforward, given the simplicity behind it. It would be inaccurate to say that he does not have a system solely because he does not have a database or that it is not digitalized yet. Vince uses a more traditional system of cataloging his inventory and tracking sales, albeit inefficient. His current system of memorizing all of his titles and tracking sales in a notebook at the register is a process he is familiar and comfortable with. Still, it has inherent limitations, namely that it is not electronic, so there is no level of autonomy behind stock counts or tracking sales figures. Additionally, there is a more significant chance for human error or data error because it relies on him entering or writing down information instead of the computer handling it. Lastly, there is no sense of redundancy, so if something happened to Vince or his notebook, all of his data would be instantly gone instead of backed up somewhere.

Operational Impact

Based on the case, it is not fully evident if there are any employees other than Vince himself or defined business segments or departments within his store. For clarity, a few assumptions will be made as documented. The first is that it is assumed there are multiple

employees with their own operational oversight, similar to a typical retail store. For example, Vince will be in charge of buying new albums and pricing them for resale, but there might be an additional employee in charge of rating album conditions, another in charge of locating customer requests, and yet another that maintains sales data. For this case, each will be considered their own operational department and may or may not have additional employees reporting to them.

Each of these business units in the store is significantly impacted by not having a fully functioning database in place in the store. First and foremost, Vince is likely at the biggest disadvantage because he has no way to make managerial decisions because he does not have any concrete data or reports that he can use for a data-driven approach. Likewise, whichever employee is responsible for bookkeeping and managing sales data will need to calculate everything by hand, a process that is time-consuming and leads to a much higher margin of error as opposed to the system calculating the data in seconds.

The employees that are tasked with managing the vinyl collection and customer requests are also limited in their daily business processes by not having a digitalized system. There is little that they can actually do currently for the employee tasked with inventory management because nothing is cataloged. There is no official way to know if the store has a specific album in stock or not. Further, the employee who manages the customer requests will be unable to tell if the store has the record already, but they also do not have a way to document the requests other than writing it on a piece of paper.

For this process and a new system to be successful, it is essential to document the previously mentioned requirements and ensure that the database system has a good design. According to Michael Hernandez, having a good design helps the database support both required and ad hoc reporting in an easy to retrieve manner, allows data to be easily manipulated, and

allows future applications to be developed much easier than a database with poor design (Hernandez, 2013). While it will be a lengthy process to import the entire existing inventory into the new database, it will likely be a significant benefit to all of the employees and the store's success moving forward.

Database Analysis and Design

Once the entire organization's current challenges, business requirements, and current limitations are understood, the database designer can start to begin developing a plan on how the database should be structured. Designing the database carefully and methodically is a critical step to ensuring success down the road when the database is finally built in the database management system (DBMS). Much like a contractor taking the time to carefully create a firm foundation for a new house or building, the design will serve as the basis for the data.

According to Microsoft, "a correct design is essential to achieving your goals in working with a database, investing the time required to learn the principles of good design makes sense. In the end, you are much more likely to end up with a database that meets your needs and can easily accommodate change" (Microsoft, n.d.). Further, there are a few components of a good database design. Microsoft notes that good design is a database that reduces redundancy through the division of subject-based tables, tables that can be joined together to report the data that a user wants and needs, thoroughly protects the information and its integrity in the database, and supports the organization's data processing and reporting needs (Microsoft, n.d.). From a design standpoint, a database designer will use a series of graphical models or diagrams to build the architecture of the system. These models include a conceptual model followed by a more in-depth logical model, and finally, the physical data model that will illustrate the final design of the database.

Conceptual Model

Conceptually, understanding the needs of Vince's business is not overly complicated. At its core, he needs a way to track his entire inventory, his customers, and all of the transactions both by customers and by his business. However, this is easier said than done when modeling it out in a complete database diagram. Conceptual models are a crucial starting point for building a solid foundation for the entire database. They offer an easily digestible interpretation of the whole database in relation to its "real world" use (Taylor, 2018). Without a conceptual model, designing the database would be much more challenging, starting with all of the intricacies like data types. The designer would likely need to go back and modify vital elements like the table structure since it was not correctly identified.

To properly build a conceptual model of the database, all of the entities need to be identified, which will encompass the key components or objects in the database. For Vince's store, the entities will be the records themselves, the customers, transactions between the customer and the store, purchases between the store and the customer, and requests for new records made by the customer. It is assumed that these five entities will be the only tables in the database in order to design this conceptual model. In theory, it would be possible to create a simple database off of these objects, but a professional database will be much more robust and complex.

After determining all of the entities in the database, the designer can begin defining all of the attributes for each. The attributes are the individual characteristics of each entity that give it meaning. Later, these attributes will be converted into separate database columns that store the data. Based on the information provided during the interviews and requirements gathering, the records entity will have attributes such as the name of the album, notes about it, and the

condition. The customer entity will contain attributes about the customer's name and phone number. The transactions and purchases entities will be very similar and have attributes about the price, tax, and total price. Lastly, the customer request table will contain information about new albums customers are interested in (Conger, 2014). Each of the entities will have a unique identifier that will be later used to identify individual rows of data.

Despite all of these attributes being listed out for each entity, there is no way for them to be directly transferred into a database without first building out relationships between the entities. For example, Vince would have all of the information about a customer he may need, but he will not have a good way to correlate that customer to a specific transaction. Likewise, he would not be able to associate that transaction with records that are being sold. It is necessary to establish individual relationships between the entities to determine how data flows through the database. For example, the designer would dictate that a transaction will pull information from the customer entity and one or more data sets from the records table to build a transaction dynamically. This is the core function of a relational database in that all of the data is always linked together through relationships (IBM Cloud Education, 2019). To make use of the information, it can be combined with values from other entities to make the results more meaningful. Data cannot be combined without building the relationship first (IBM Cloud Education, 2019).

Figure 1 below shows a basic conceptual model with all of the entities listed and their respective attributes. In this case, the entities are the actual vinyl records, the customers, their requests, the transactions made by customers, and the purchases made by the business from customers. The model highlights some of the critical relationships in the database, such as how customers are associated with transactions, purchases, and customer requests and how records

are connected to transactions and purchases both made by Vince and by the customers. The actual data model, shown later in Figure 4, will contain several more relationships than this that are more robust, but this model demonstrates how the business requirements can be met from a high-level overview of the case. The intent of the conceptual model is to provide a quick overview of the database structure from a broad level.

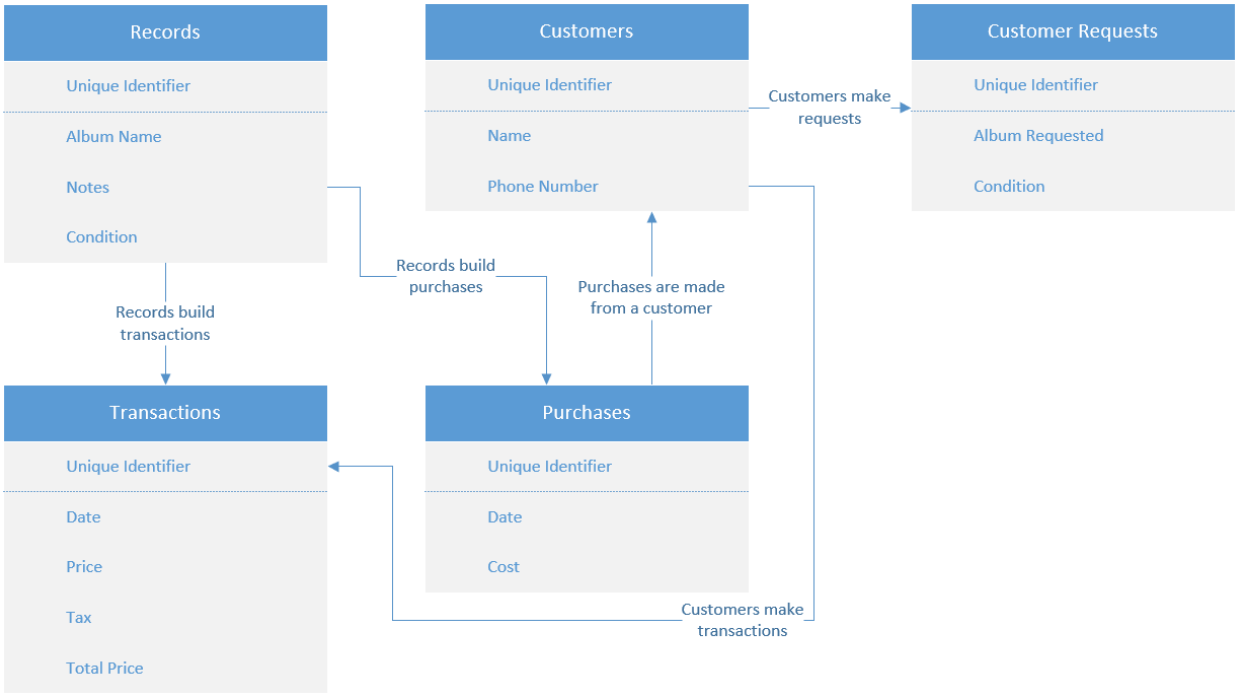


Figure 1 This basic conceptual diagram created by the author shows the five entities of records, customers, customer requests, transactions, and purchases. Each entity has a unique identifier and several attributes that give characteristics to the entity.

As previously mentioned, the attributes and entities listed here would be satisfactory to meet Vince's needs and the business rules previously mentioned; however, additional attributes can be added to each of these entities to increase the overall functionality and usefulness of the database. For example, it would be ideal for Vince to track key customer information such as their address and email address if Vince wants to send marketing materials or flyers to them. Additionally, the records entity is very simplistic while it is the core of his business. Expanding

to include additional attributes like the artist and year would have better reporting to help customers find what they are looking for.

Logical Model

After designing a conceptual data model, the database designer can begin to create a logical model. The logical model is a more in-depth view of the entire project in terms of the individual relationships and all of the attributes listed in their final form. While a logical model is self-contained for the whole project, the models can integrate with other logical models if the project needs to integrate with other systems or applications (Conger, 2014). While no primary and foreign keys will be identified at this level, the entities will begin to convert to their final table form, and the relationships will start showing the cardinality based on how the data is related between tables. For example, one and only one customer can be associated with a specific transaction, but a customer can have zero or many individual transactions associated with them. For this project, crow's foot notation will be used, which shows zero, one, or many relationships such as zero-or-one, one-and-only-one, zero-to-many, or one-to-many. Figure 2 below shows the common symbols associated with crow's foot notation.

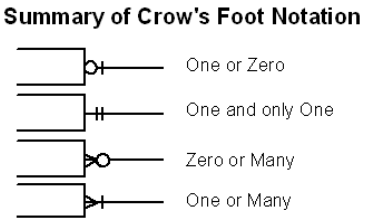


Figure 2 This diagram shows the traditional symbols of crow's foot notation in database diagrams. Each end of the relationship will display these symbols to denote the relationship's cardinality between the tables (Stewart, 2008).

In addition, during the logical modeling process, it is ideal to begin adding linking and lookup tables, where necessary, to support the overall design and function of the database. According to Conger, linking tables can be used to resolve many-to-many relationships into one-

to-many relationships (Conger, 2014). In other words, the database currently has a table for transactions and another for customers. Still, it requires a linking table to begin associating specific transactions with customers and vice versa. Additionally, during this time, it will be essential to start adding lookup tables that are more of a dictionary than a traditional database table that will constantly modify data (Conger, 2014). For example, instead of storing each condition name for each record in the inventory, it would be best to reference a specific condition ID in a lookup table so that if a condition name needs to change in the future, it only needs to be updated in one spot instead of for each record.

In the design of this logical model, a number of these new linking and lookup tables were added to support the overall structure of the database after all entities were converted to tables in their own name. New linking tables include ItemsToTransactions and ItemsToPurchases, where new lookup tables have been added for the conditions, the type of record (LP vs. 45 vs. 75 RPM), and the discounts given. Additionally, the records table has been split into two separate tables to reduce redundancy in the database. Instead of needing to have the same album information repeated multiple times for each record type and condition, a possibility of 12 separate times, the new inventory table will join with the lookup tables to identify a condition and type while storing pricing information. The existing records table will hold important record-specific information like the title and artist since it is the same information regardless of the specific copy in stock.

Further, several additional attributes have been added, as previously mentioned. These other characteristics about each entity will lead to easier user interaction on Vince's and his employee’s parts and make the entire database more useful for his day-to-day business. It was identified during the requirements gathering that he does not have a way to aggregate data or pull

profit reports, so these new data points should help him achieve that. Lastly, other relationships were added to the diagram to show all possible connections between the individual tables.

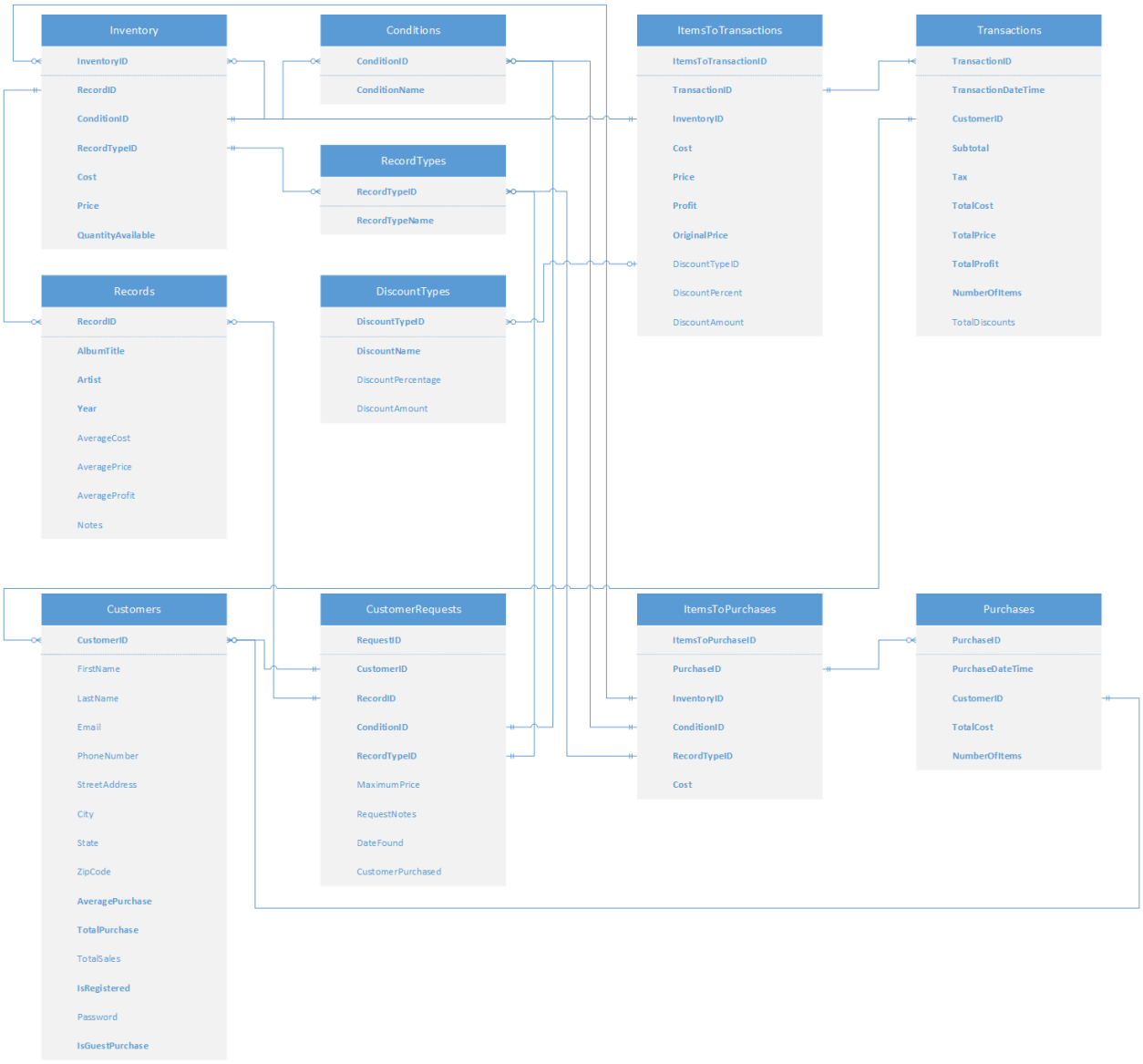


Figure 3 This logical model created by the author shows all of the identified tables, their attributes, and relationships in the database. The individual relationships have the cardinality noted in crow's foot notation. Required attributes are indicated in boldface text.

Physical Design

Once a designer finishes a logical data model for the database, and it has been approved, the final physical data model can be constructed based on the finalized design. The physical data

model is a much more comprehensive diagram showing nearly everything regarding the database structure. Similar to the other two models, it will show all of the attributes and their relationships, but it will also expand this information by showing the primary and foreign keys used to build the individual relationships (Taylor, 2018). While a physical data model is more encompassing and contains more detail, it is not always the easiest to manage compared to the other models. Any slight change to the model's layout, even changing a data type, can result in significant changes and restructuring of the diagram (Taylor, 2018). Given this, it is essential to complete the models sequentially to minimize the number of changes at the end of the process. Ideally, this diagram should have little to no changes and can be used directly to create the DBMS.

For Vince's database, the approach taken to finalize the data model was to use three separate schemas for the records, the customers, and the transactions. While using different schemas as opposed to the default ‘dbo’ schema in SQL Server Management Studio (SSMS) likely is not necessary for a simple database of this size, it is still good practice compared to a more robust, professional database and does come with some advantages should the need arise in the future. First and foremost, using separate schemas provides a logical separation in the database (Gupta, 2021). For example, the ‘transactions’ schema will only house tables associated directly with transaction information such as the purchase, discounts, and transactions themselves as well as the linking tables. Similarly, the ‘customers’ schema contains customers information, and the ‘records’ schema holds all tables directly involved with the inventory.

Beyond this, differentiating schemas allow for more advanced control and security of the database. Instead of needing to set permissions and access to users and objects on the global or table levels, the DBMS administrator would be able to grant or revoke access based on the

schema (Gupta, 2021). This is particularly helpful if Vince plans to add additional applications or interfaces in the future that will connect to the database. If he has a web portal that shows the current inventory, that page does not need access to the ‘customer’ or ‘transaction’ schemas, only the ‘records’ schema. This will also allow future updates to the schemas to be much more streamlined than having a common default schema.

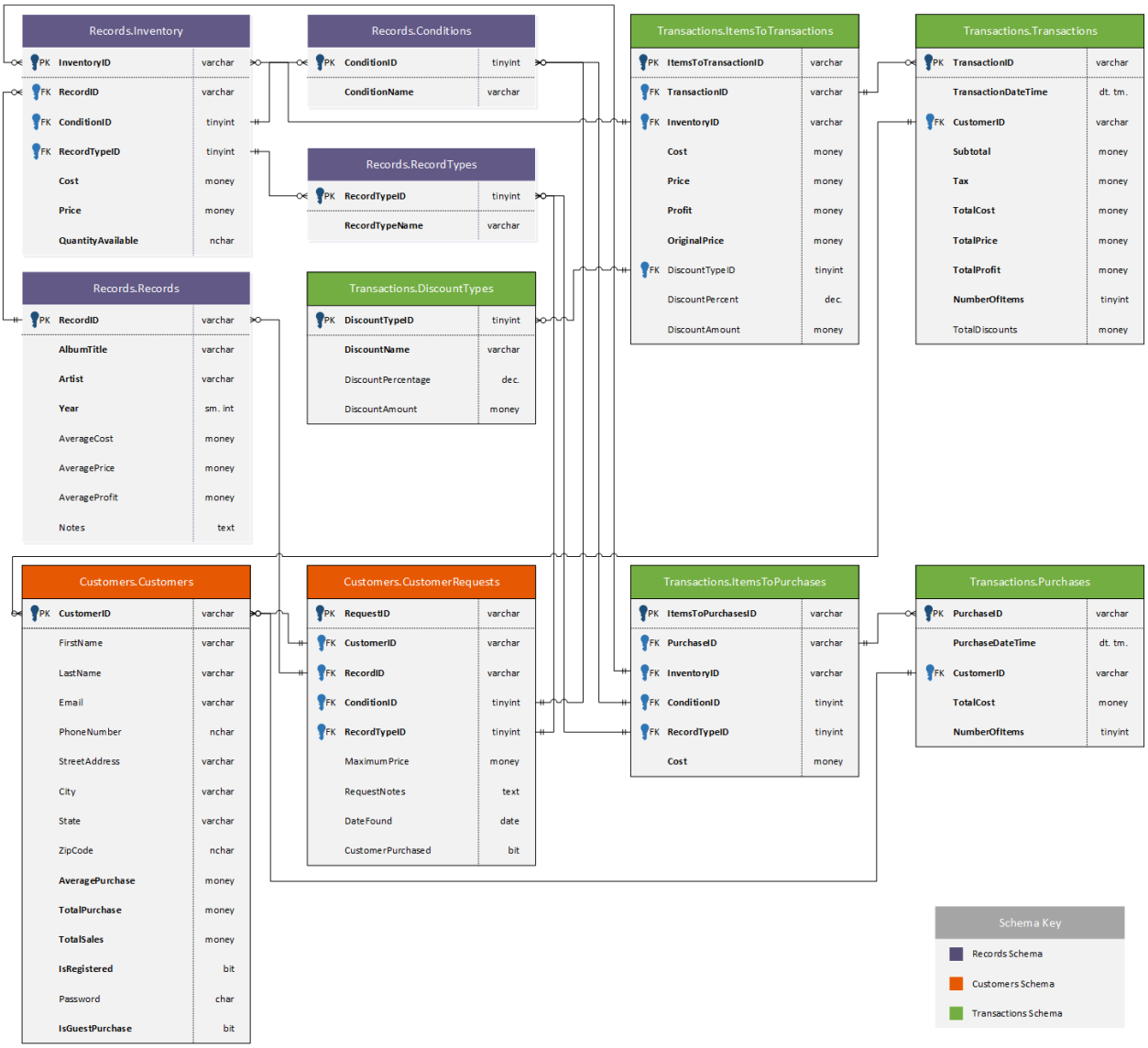


Figure 4 This physical data model created by the author shows all of the respective tables in Vince's database. The database is comprised of three unique schemas for the records, customers, and transactions. Each table has several attributes noted with their given data types. Required attributes are indicated in boldface text, and relationships are shown with crow's foot notation cardinality.

Using the database model shown in Figure 4, sample queries and scripts are provided in Appendices A through C. Appendix A provides various reporting queries to report on key metrics that Vince cannot currently do. Appendix B includes a number of stored procedures and executable statements that show how data can be updated in the database. Finally, Appendix C provides the complete database creation script that will create a new database following the data model as well as seed the tables with the sample data provided by the case text.

Database Management Systems

When designing a new database implementation for any scenario, it is crucial to the overall success to pick a database management system that will adequately support and maintain the database being built. As important as it is to have a robust set of features for the technology running in the background, it is equally important to consider the user interface of the users maintaining it and any possible integration and configurations needed. No matter how simple or complex a database design may be, its implementation hinges on properly selecting its database management system.

Research and Analysis

The database management system (DBMS) and the database engine are the powerhouse of the entire database that runs, updates, and retrieves all of the information. As with any system or application, several different options and vendors are available at all different price points. Likewise, each touts its own set of features and contains unique disadvantages that might hurt the implementation at Vince's store (Conger, 2014). This comparison will evaluate four major systems, Amazon Web Services' Aurora platform, Microsoft's SQL Server, PostgreSQL, and MySQL.

Amazon Aurora

Backed by the ever-growing tech giant, the Amazon Aurora database engine is one component of the more extensive Amazon Relational Database Service (Amazon RDS) serviced through Amazon Web Services (AWS, 2019). Amazon Web Services (AWS) is a complete development solution that features numerous types of database services like Aurora and over 100 other products ranging from networking to game development to web hosting. Like all of the other AWS products, Amazon RDS using the Aurora engine is entirely cloud-based, unlike every other option on this list which is on-premises (AWS, 2019). This alone gives a distinct advantage for Aurora as additional hardware and software are not needed upfront.

Amazon Aurora is a fully managed database and is widely used worldwide, given it is a part of the AWS ecosystem. The system's main advantage is that it is truly integrated with all of the other products, which could be very helpful for Vince if he'd start hosting a website, online portal, or other applications. Additionally, the system is highly scalable. Its price is based on the size and usage, so Vince would not be overpaying for features he is not using; rather, as his business grows and he needs more resources, he will only pay for it them at that time. Further, Amazon RDS has built-in, user-friendly performance monitoring, so Vince would easily be able to keep a close watch on his database without the need for expensive hardware and software, despite not having a technical background (AWS, 2019).

However, Amazon Aurora does have a few crucial weaknesses to consider. While it is all priced based on how much data is used and how often the tables are being accessed, it can be costly for high resource-consuming queries or applications. Charges can be a matter of cents when the database is active. Still, users are billed for a minimum of five minutes of availability despite being live for potentially a few seconds. Additionally, since everything is based on

separate cloud instances, it would be cheaper to host on a smaller instance, resulting in fewer features, forcing users to pay more for larger instances with more features available to them (AWS, 2019).

For Vince's store, this would likely be an excellent option for him. Amazon Aurora is a reliable solution with many essential integrations that he could potentially use in the future, such as hosting a website. Given that everything connects within the AWS family of products, similar applications will have a standard look and feel that would allow him to make changes more quickly and navigate the websites to update what he needs. Further, this is a cloud-based option that would allow him to save money upfront from needing to buy additional hardware. However, the cost structure could become expensive for him over time with his growing business as all of the products under the AWS suite have volume-based pricing instead of just a flat fee or license fee (AWS, 2019).

Microsoft SQL Server

Trusted and used widely for years, Microsoft's SQL Server platform has been implemented in a variety of scenarios, from small businesses like Vince's to large, global enterprises. The on-premises server is license-based with applicable options for this case, starting at $1,000. SQL Server has always been known for its reliability and advanced feature set, allowing programmers and database administrators to configure the server in countless ways to achieve exactly what they need from it for any number of programming languages (Microsoft, 2019). Since it is backed by Microsoft, there is naturally a wide array of resources available as well as numerous support forums. In terms of reporting, SQL Server is meant to integrate directly with applications like Power BI that allow for advanced report dashboards for metrics like sales information (Microsoft, 2019).

Microsoft SQL Server is not as user-friendly as its other applications like Word or PowerPoint. Given the nature of setting up a server in general, most users would likely find this product to be rather complex or hard to manage, partly due to the level of customization that can be achieved. Likewise, the licensing can be fairly confusing as there are multiple levels, each with different features available (Microsoft, 2019). Microsoft is constantly adjusting these tiers as well, which can make understanding them even more difficult.

Vince could potentially implement SQL Server for his needs in his Vinyl Shop. Out of all of the options, it is the one that is more "elite" within the data industry and is highly regarded as being a forerunner. Despite the complexities and configurations surrounding the system, Vince likely would be able to use the software to meet his needs and track sales and inventory data. SQL Server integrates well with Power BI, which could be a major benefit for him as he works on figuring out his profits and sales reporting.

PostgreSQL

Unlike Amazon Aurora and Microsoft SQL Server, PostgreSQL is completely open-source and has origins dating back to 1986 when the POSTGRES project was implemented at the University of California at Berkeley. Being open-source and maintained by different individuals around the world, there is an inherent concern for reliability and security. Despite being 100% free and community-based, PostgreSQL conforms to almost every SQL:2016 Core conformance standard, works on every major operating system, and has its own supported library of integrations and add-ons that make it even more desirable to developers. Since it is managed by everyday users, the documentation available on the system is quite extensive compared to other systems (PostgreSQL, 2019).

However, given PostgreSQL is free and open-source, there is limited support available for users since no company owns the application. If issues arise, users need to rely on documentation and forums solely instead of being able to call into a support line. Additionally, compared to the other systems, PostgreSQL tends to have lower read speeds and latency which could become problematic for large, complex queries or data reports that Vince would need to run for his business. Similarly, if he were to have a list of his inventory online, a storefront website, for example, he would need to ensure fast speeds for the sake of customers being willing to stay on his website.

PostgreSQL would be a viable option for Vince as it is the cheapest option, given that it is completely free. He most likely would need to purchase a server to run the software on, but he will not be tied to any yearly fees or licenses. PostgreSQL is a widely trusted database system as it has been around for more than three decades and has a host of resources available. While he would not have a direct customer support line or email that he could reach out to, there are a number of integrations that he could benefit from, plus he has all of the online community for support and assistance if needed.

MySQL

Similar to PostgreSQL, MySQL is an on-premises open-source solution that could potentially work for Vince's business. MySQL is currently managed by Oracle and is trusted by a number of major companies like YouTube, PayPal, LinkedIn, and Facebook, given its wide integration with hundreds of major platforms and applications. The software is a component of the widely popular LAMP stack (Linux, Apache, MySQL, Perl/PHP/Python) which has partly allowed it to become so successful over the years as most Linux developers utilize LAMP

software. Depending on the implementation being used, Vince might be able to use the database engine for free (MySQL, 2000).

In terms of limitations, MySQL's licensing can be very confusing. The core software is open-source and is available for public download for free under the GNU general public license, but this does offer a much smaller set of features. For full features, users would be required to pay an expensive licensing fee every year. Further, like Microsoft's SQL Server, there are multiple licensing levels, each with different features and components, which can make determining which to purchase rather confusing (MySQL, 2000).

MySQL could work for Vince's vinyl collection and sales either as a free version or under one of the full licenses. While the free version would come with fewer features, it would save him more money compared to the licensed versions. However, he would still need to purchase additional hardware to support the system. A full comparison of each of the mentioned database management systems can be seen as follows in Table 1. The table includes the license types, advantages, disadvantages, and other general information about the systems.

Criteria	**Amazon Aurora**[1]	**Microsoft SQL Server**[2]	**PostgreSQL**[3]	**MySQL**[4]
License Type	Proprietary	Proprietary	Open Source	Open Source
Implementation	Cloud-based	On-premises	On-premises	On-premises
Price	Scalable based on use	Starting at $1,000	Free	Free for Open Source, Starting at $2,000/year for Integration
Advantages	Easily managed and simple to use interfaces, scalable for future growth, performance monitoring, fully managed	Very reliable, works with a number of development languages, industry leader, backed by Microsoft	Free, widely supported and maintained around the world, numerous forums and documentation sites, scalable	Free for base implementation, simple and easy to use, widely supported by a number of applications and services
Disadvantages	Can be very expensive for apps consuming a lot of resources, small	Full features are only available on top licenses, licenses are expensive and constantly changing	Low read speeds compared to other systems, not as widely supported as	Complex pricing and licensing, not efficient with large datasets, more prone

	instances have limited resources		MySQL for being open source	to data corruption than others
Integrations	Integrates in seconds with all of Amazon Web Services (AWS)	Integrates with Power BI and SQL Server Reporting Services for data reporting	Integrates with a variety of open source applications that are also free	Integrates with hundreds of applications for reporting and marketing
Current Customers	Samsung, DoorDash, Pokemon, Dow Jones	Microsoft, Stack Exchange, Alibaba	Uber, Instagram, Spotify, Robinhood	YouTube, PayPal, LinkedIn, Facebook

Table 1 This table created by the author shows the general advantages and disadvantages between the referenced database management systems as well as information about them. [1](AWS, 2019), [2](Microsoft, 2019), [3](PostgreSQL, 2019), [4](MySQL, 2000)

Recommendation

For Vince's vinyl store, utilizing Amazon Web Services' RDS running the Amazon Aurora engine for his database management system would be recommended. First and foremost, Amazon Web Services is completely cloud-based, meaning Vince would not require any specific hardware to run the databases, and he would have the ability to log in and manage his data and services from virtually anywhere, on any device. If he chose SQL Server or one of the open-source systems, he would need to buy an expensive server or computer. If he wanted to access the data from outside of his business, at home, for example, he would either need to open his server to outside connections, which creates a large vulnerability, or he would need to purchase an expensive virtual private network (VPN) software.

Further, choosing to go with Amazon Aurora would allow Vince to take advantage of the rest of AWS in terms of web hosting, data reporting, and marketing applications that he currently does not have access to. This would be a complete technology solution for him to completely change the way he does business and interacts with his customers. Compared to other options, he would be able to save a significant amount of money because he will only need to pay for what he uses from AWS. Lastly, if he does find that this package works, he will not have invested any

additional money into hardware, but if it does not work, he can simply stop using the service without being locked into additional fees or licenses.

Hardware and Software Support

As previously mentioned, selecting the Amazon RDS system running Amazon Aurora would allow Vince to avoid needing to purchase any additional hardware. This assumes he already has a computer or device that will act as a register and connect to the system. In terms of software, Vince will need a way to connect to his databases. He likely will not directly input data into his database using the interface online, nor should he be doing this as that is a bad practice that can lead to data integrity issues from human error. To facilitate this, he will need a user interface to facilitate this process. The interface can either be custom-built for his needs or can be an interface that is generic and purchased as a proof of concept until he has more of a need to have one built.

A simple way to have an interface to use to input sales data, manage his inventory, and run reporting would be to use other Amazon Web Services products like AWS AppSync, and AWS Amplify to create a simple website that would connect to his AWS Aurora database (AWS, 2019). While it would be time-consuming to do so, he would be able to adapt the website to allow customers to log in and make purchases right from his website, which they cannot currently do. Alternatively, albeit more difficult for him, he could use a language like JavaScript to build his own GUI interface. This would not be very feasible for someone without development experience, but there are countless resources available to help create it. However he decides to go about it, choosing Amazon Aurora for his database would likely save him the most money in the long run.

Data Model

A vital component of the database design process, enterprise data models help ensure the overall understanding and success of a database by establishing how data is related to other data and why it exists in the manner that it does. For example, it would be possible for Vince to work completely out of Excel spreadsheets for all of his business functions, but at some point, he will run into issues where data is mismatched or does not correlate as expected. Having a proper data model upfront can enforce data logic that matches existing business rules right from the beginning.

Enterprise Data Model

Enterprise data models are a carefully balanced set of visual representations of data in an organization (Kendle, 2005). The data can be presented in various mediums such as diagrams or charts as long as it adequately explains the data from a business and operations perspective. "It focuses on high-level, more abstract components as it tries to define and standardize an entire enterprise business' data" (Bennett, n.d.). While creating a complete data model of this nature for an entire enterprise-level database would be extremely time-consuming, there are significant benefits, including verifying that the data can remain clear of redundancies and inaccuracies, ensuring it will be easier to manage and maintain, and ensuring it will remain more secure given there is a logical ordering for everything (Bennett, n.d.). The enterprise data model will be closely tied to and based on the physical data model already created. However, the physical model might only show a subset of the database.

While some organizations may have different components for their data model, including data dictionaries and XML schemas traditionally, an enterprise data model will include three core components. Those components include a subject area model, a conceptual model, and an

entity model (Kendle, 2005). The complete model itself can represent the entire organization, or it can be specified in some manner to a specific department or group within the company. If it is group-specific, it should still reflect and compliment the operating rules of the collective organization. While the case study does not specifically mention specific departments in Vince's store, it can be assumed that if he expands, there will likely be multiple departments or roles such as sales, inventory management, and customer service, among others. For this data model, the inventory management department will be used, which oversees all of the current stock and adjustments for sales and purchases.

Subject Area Model

In the enterprise data model, the subject area model is both the highest-level diagram and the least technical. The model shows a very basic overview of the company's data and how it relates to each other across the database, much like the conceptual model created and shown in Figure 1 (Bennett, n.d.). Within the subject area model, any entity of significance such as a department, user, or resource is identified similarly as a subject area and is connected through existing or abstract business processes. At this stage, it is not necessary to note these entities as tangible items versus resources or users in the model because they still require the same level of interaction with each other to have data flow properly. Likewise, the relationships that connect the subject areas might not be the same relationships that exist in the final database, but they should be included if they indicate how the subject areas are connected. These connections will later be refined in future models and iterations.

In the case of Vince's store, each of the departments mentioned above are their own subject area, as well as the customers and the inventory. It is important to identify early on and set the expectation now that customers, sales, purchasing, and any other department will not

interact directly with the inventory; rather, they will receive information through the inventory management team. This adds a level of protection and integrity against the data so that only one subject area is editing that data. Otherwise, there is an increased potential for inaccuracies. This model, while very simple, is a crucial step to help ensure data is not duplicated throughout the database. For example, it is acceptable for the inventory to manage the name of a record in Vince's collection, but it is not necessary for the sales department to also manage a copy of this. The model shown in Figure 5 shows the proper flow from inventory to inventory management to sales should they need access to that information.

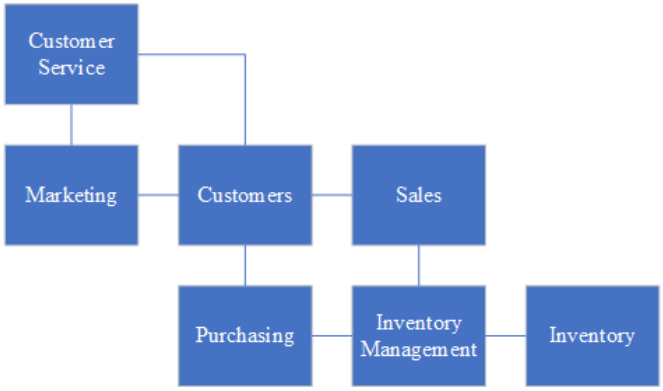


Figure 5 This subject area model created by the author shows an abbreviated view of the store's data interactions. Customers interact directly with customer service, marketing, sales, and purchasing. Only inventory management can update the inventory itself from their interaction with sales and purchasing.

Conceptual Model

After creating a subject area model, the conceptual model is created in a similar fashion but with an enhanced and more in-depth view of the data. This diagram specifically starts to highlight all of the particular business rules that need to be abided by when architecting the design of the database. For example, without having a thorough understanding of which department or application is updating stock versus communicating with customers, it would be impossible for the designer to build a proper design adequately. It is likely that there would be

inefficiencies or errors in the data at some point as a result because the data is not being stored in an efficient manner.

According to Kendle, these models are designed to help verify the scope of each subject area along with how they are connected. The model shows through a series of relationships and connections or even just a concept how data is transmitted through the business (Kendle, 2005). For example, it was previously identified that the customers subject area would not directly interact with the inventory data. Rather, customers will make a purchase from the sales team, who will, in turn, pull the database record from the inventory subject area or at least notify them. The inventory management team will receive the request who will then update the stock in its database tables. Naturally, this will not be a manual process by a single user, rather it will be automated when a cashier scans a record onto a sale, but the data is still controlled by the inventory management team. Given the digital nature of the database, updates should always be made through API calls or stored procedures as opposed to a user making manual updates. Doing so will add a more logical structure to the database along with ensuring the correct data is being updated and isolated to only what is necessary to update (Wagner, 2019).

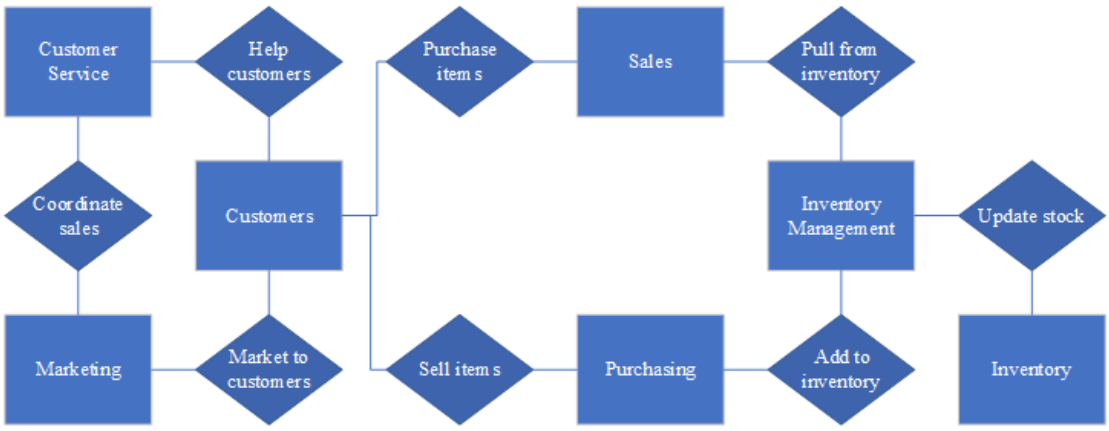


Figure 6 This conceptual model created by the author shows an enhanced view of the flow of data. The diagram highlights all of the major concepts and business rules, such as how customers buy and sell records and how the stock is updated for the inventory itself.

Entity Model

After the conceptual model is created and accepted, database designers can expand the subject areas to build the final component being the entity models. The model is typically in the form of an entity-relationship diagram (ERD) and highlights individual tables and attributes within the subject areas (Bennett, n.d.). While ERDs can encompass the entire database structure with all tables, it is also common to see them only contain a subset of information and tables, so it is easier to understand the area's concepts (Kendle, 2005). Depending on the size of the database and the concepts being explained, it can be confusing for the user to see the entire database instead of a small section of it related to what they need to know. For example, it is not fully necessary for a designer to understand the ‘customer’ schema for Vince's table when the inventory management team only needs to work in the ‘records’ schema.

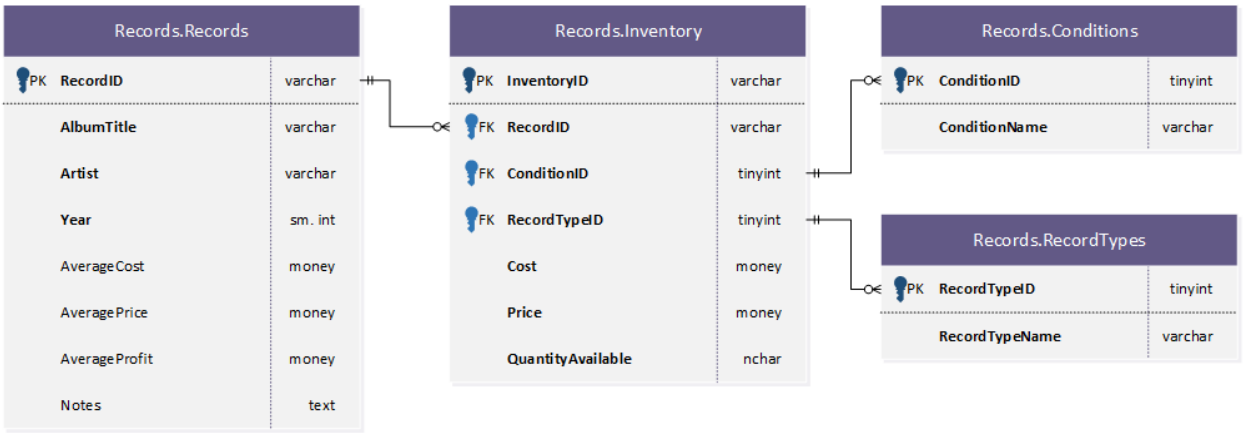


Figure 7 This entity model created by the author shows the four tables in the ‘records’ schema of Vince's database. The diagram highlights the relationships between the tables as well as the attributes and data types of the columns in each. This schema is only a subset of the entire database but encompasses the relevant tables for the inventory management team.

Operating Rules

Business and operating rules are a crucial component of any software development process as they "provide the foundation for automation systems by taking documented or undocumented information and translating it into various conditional statements" (IBM Cloud

Education, 2021). Essentially, the software, in this case a database system, should complement the existing processes of the organization to help expand and automate them instead of negatively impacting them. The operating rules should encompass the team or department in question and be concise enough to understand and reflect in the code.

An important distinction should be made when beginning to identify all of the operating rules of the inventory management department. While the physical team is in charge of all of the inventory, other stakeholders such as customers or cashiers will also interact with the data, just through the team itself as a proxy. As such, several of the business operating rules that have been identified do cross over into other domains but still are focused directly on the inventory. Table 2 below shows the current operating rules that have been identified during the organizational analysis for managing the store's inventory. There are several basic rules, such as viewing the current inventory and managing the stock levels, as well as customer-facing rules, such as associating products with transactions and purchases and managing customer requests. Additional rules listed in the table include adding to, editing, and deleting from the conditions and record type lookup tables.

Business Rule	**Query Type**	**Affected Tables**
View the current inventory	SELECT	Records.Inventory with FK to Records.Records, Records.Conditions, Records.RecordTypes
Edit the quantity available	UPDATE	Records.Inventory
Associate an inventory item with a transaction	SELECT, INSERT	Transactions.ItemsToTransactions with FK to Records.Inventory
Associate an inventory item with a purchase	SELECT, INSERT	Transactions.ItemsToPurchases with FK to Records.Inventory
Request a new record	SELECT, INSERT	Customers.CustomerRequests with FK to Customers.Customers, Records.Records, Records.Conditions, Records.RecordTypes
View existing requests	SELECT	Customers.CustomerRequests
Edit existing requests	UPDATE	Customers.CustomerRequests with FK to Customers.Customers, Records.Records, Records.Conditions, Records.RecordTypes
Delete existing requests	DELETE	Customers.CustomerRequests
Add a new inventory	INSERT	Records.Inventory with FK to Records.Records, Records.Conditions, Records.RecordTypes

Edit existing inventory	UPDATE	Records.Inventory with FK to Records.Records, Records.Conditions, Records.RecordTypes
Delete existing inventory	DELETE	Records.Inventory with FK to Records.Records, Records.Conditions, Records.RecordTypes
Add a new condition type	INSERT	Records.Conditions
Edit an existing condition type	UPDATE	Records.Conditions
Delete an existing condition type	DELETE	Records.Conditions
Add a new record type	INSERT	Records.RecordTypes
Edit an existing record type	UPDATE	Records.RecordTypes
Delete an existing record type	DELETE	Records.RecordTypes

Table 2 This table created by the author shows all of the business operating rules related to inventory management. Most of these rules would be completed automatically through queries instead of being manually completed by the inventory management team, but it is still under their control. Note: The table listed first for each rule will be the primary table affected and any other table will be accessed through a foreign key (FK) relationship and JOIN statements.

Rule Reflection

Once all of the data models have been constructed and approved by the entire team of stakeholders and the operating rules have been identified, it is important to determine if the data models actually reflect the rules properly. Without it, it cannot be guaranteed that the new database will support the business's functions properly and it potentially could be more of a hindrance than anything. After review, the models that have been created for Vince's business do complement the operating rules and should lend themselves for a complete database design to support his business.

First and foremost, it was determined during the analysis and design phase early on that the overall physical design would utilize three unique schemas as opposed to all of the tables being placed in the default ‘dbo’ schema. Not only will this help with security and permissions later on, but it will greatly help to separate tables in a logical manner. In this case, the inventory-related tables will all be in the ‘records’ schema, similar to how inventory management will be its own subject area in the data model. Lastly, the models themselves support the business rules

for inventory management in terms of how data is moved through the database between tables. Just like the subject areas mentioned above crossing over and acting as conduits between each other for the data, the proposed entities and tables will have similar relationships through foreign keys that will connect data points while eliminating redundancy.

It is also important to note that the proposed data models referenced support not only the inventory management team's operating rules but also the business as a whole. Vince currently does not have an electronic way to track his sales, inventory, or customers, so anything should be an improvement, but this structure should be malleable enough to get him started while being scalable as the business grows in the future. The database allows him, his staff, and his customers to browse the current inventory selections, associate stock with transactions and purchases, request new records, and run various reporting metrics that he cannot currently do. Paired with a user-friendly interface, this model should help further his business operations significantly.

Law, Ethics, and Security

No matter how good the design or structure of a database is, if the designer does not take into consideration legal compliance or ethics, the database could be susceptible to a number of issues from fines to loss of business. In general, this database implementation is fairly straightforward and does not have the same level of compliance needed from a database that would be used in education or healthcare, but that does not mean it does not have to follow suit in some manner. Though the need to comply will vary between the departments in Vince’s business based on the data they store, there will be common themes throughout the entire store’s database. For the purposes of clarity, the following will primarily focus on the inventory management team.

Standards

According to information scientist and executive Benjamin Alexander De Mers, “the only defense against the unethical use of information is the ethical standards of the stakeholders themselves” (De Mers, 2014). Despite long-standing advancements in technology and surrounding laws, such as the General Data Protection Regulation (GDPR) in Europe, the United States does not have any overarching data protection laws to date (Klosowski, 2021). As such, the biggest concern that will affect the design and implementation of Vince’s database will be surrounding various ethical standards. However, depending on the states that the store will be conducting business in or where the store is actually located, there may be additional state-level or local laws to consider.

By far, the most significant standard for Vince’s store is the Payment Card Industry Data Security Standard (PCI DSS). Created by a joint coalition by American Express, Discover, JCB, Mastercard, and Visa, the PCI DSS is a set of rigorous instructions for everything from maintaining firewalls and other security measures to how applications and this database would be managed (Berecki, 2019). These standards cannot officially be enforced at a legal level but have been widely accepted as the best ethical practice to follow for retail stores and other industries that use payment cards for transactions. While this database will not store payment card information, Vince’s store will still process transactions, and the same standards would apply to the database system.

Further, Vince would likely be affected by state regulations such as the California Consumer Privacy Act (CCPA) if he is located in California or sells items there, such as if he started selling online. The CCPA was a landmark act geared toward increasing consumers' rights instead of the businesses collecting customer data. Under the law, consumers can request to have

their information deleted from a system and must be properly notified of how their data is being collected. From a business perspective, Vince's store would be liable for properly securing customer data (Berecki, 2019). Additionally, it is worth noting that the store could technically be accountable under the Federal Trade Commission's Act against deceptive practices, albeit it is less relevant in this scenario for the database (Klosowski, 2021).

Legal Compliance

In a recent article by Thorin Klosowski, the editor of privacy and security topics at Wirecutter, it is noted that data experts agree focusing on four common areas can help to ensure data protection, and by virtue, legal compliance with relevant laws and legislation. Those four areas include data collection and sharing rights, opt-in consent, data minimization, and nondiscrimination and no data use discrimination (Klosowski, 2021). In terms of data collection and sharing rights, customers should always be able to readily see what a business is collecting on them in terms of data. For example, Vince's customers should be able to see he is not only collecting their names but also their addresses, phone numbers, and email addresses. Likewise, customers should need to opt-in to data collection instead of it being an automatic process and needing to opt-out later once they realize it. For his store, this could be as simple as having a notice on the future website when customers register for a new account or a verbal notification when a customer provides their phone number when making a transaction.

Data minimization more so focuses on the design of the database. This standard surrounds similar logic to "less is more," meaning a company should only collect the data they need; there is never a need to collect everything possible just because they are able to. For this database, it is specifically designed only to collect the information that Vince would need to contact customers for promotions or flyers or if he were to mail something to them. Lastly,

nondiscrimination surrounding data use is more of a business rule than a database design concern. Essentially it states that the company and applications should never discriminate against a customer that exercises their rights to opt-out of data collection or wants their data to be removed from the database.

While following these standards will not exclusively mean that Vince's store is fully compliant from a legal perspective, it can significantly help to achieve compliance. As mentioned, laws will vary between states and are constantly changing. However, these standards are more proactive than most of the laws are to date. Following them will mean that Vince's store and the database and applications are going above and beyond the minimum requirements of his jurisdiction.

Ethical Practice

Beyond legal compliance, database designers also need to consider any of the ethical practices that are standard in the industry as they relate to the database design, storage of data, and data use. Ensuring that proper practices are in place will help mitigate potential data loss, corruption, and theft. While this is never guaranteed, it can at least add additional safeguards and peace of mind along with all of the other compliance issues and standards previously mentioned for the database.

Ensuring an ethical design of the database can be argued as being one of the single most important practices for a database designer to observe. Without having a proper design, there is an inherent risk to the security of the data as it is not structured to protect the information, or it might be structured in a way that makes the management of the database difficult for users. As mentioned, this database is designed to be segmented into three separate schemas for the records, the customers, and all of the transactions both to customers and from customers. From a

structural perspective, having all of the inventory management data in its own schema protects it as it will be clearer to users where their data lives when working on it in the future, as well as for assigning access.

Despite how far technology has come in recent years, it is never going to be fully immune to data loss and corruption from hardware malfunctions. The database technically could be fault-tolerant through redundancy, meaning it will be able to continue when a component fails, but there is always a risk associated. For the inventory management team, this could be a major issue if they were to lose all of their data on the current stock. This could potentially set them back weeks by needing to go back and re-inventory the entire collection. To combat this, the database should regularly be backed up to either physical media or to the cloud for easier access and reliability. However it is stored, it is everyone's responsibility to ensure that the backups are stored just as securely as the database itself. For example, if it is backed up to a portable hard drive, that device should be secured in a safe or locked area instead of just laying out on Vince's desk.

Ethical practices must also be observed when considering how the data is going to be used. For example, Vince will have access to customers' names, addresses, phone numbers, and email addresses. This information is protected information that he is directly responsible for its proper use. He and his employees should only be using the information in the database for their business functions, for example, sending coupons to customers. However, there is a fine line between sending flyers and promotions to overloading a customer's email with multiple emails a day. This is something that is a question of ethical practice as opposed to official legislation that he must follow. In terms of inventory management, it would be much harder to misuse the data

ethically, but it can still happen if an employee were to steal the information or deliberately share incorrect information with customers.

Security Needs of Solution

The minimum level of security or the security needs of a database will always vary by industry as some industries such as healthcare, banking, or education will likely have heightened needs compared to a local store selling records. However, ensuring the minimum needs are not only met but are exceeded is an important step to ensuring proper design and successful implementation that is compliant with industry standards. The needs of individual stakeholders or departments within the company can vary as well based on the data they are able to access and what their use of the data is. However, given the nature of this retail store, all needs can be summarized into the data remaining secure, private, and unchanged from unauthorized users. For example, a customer would need their private information, such as their address, to remain secured just as much as the inventory management employees would need information about records to be safe from data loss or intentional manipulation by intruders.

Specifically looking at the inventory management team, it is likely that there would be concerns related to the stock counts for individual items as this is the center of their team. If a hacker would update stock counts to be higher than they are, the store could easily oversell items. Likewise, if they were to lower the prices and then buy a large number of items, it could cost the company a significant loss. Data manipulation is not only subject to external entities such as hackers. It is possible for a bug to exist in the code, such as a frontend interface, that could update the quantity or price for items without anyone even noticing. A cashier performing the transaction could simply think the price was correct if they noticed it was different, and the bug could continue to update information.

Database Security Plan

Having a proper security plan in place for the database is a crucial step in maintaining the integrity of the data, as well as ensuring that if the hardware does fail, the data can be recovered. The data plan provided is based on current best practices but should constantly be reviewed and modified as needed to meet the needs of Vince's changing business. For example, the plan may need to be expanded and become more robust as his business grows in the future, along with the growing size of the database.

Authentication:

Given the nature of having both internal employees and external customers accessing data from the system, it will not be as simple as using the default Windows Authentication for the server. Instead, it will be a mixed approach. Internal employees will be required to access the servers by a designated username and password assigned only to them as they will likely be sharing a register. When it is their turn to use the register, they will just log in with their credentials. For customers, they will not have direct access to the server; rather, they will need to connect through a web interface. To do this, they will log into the website, a certificate will authenticate them, and the system will log them into the server. Without the certificate, they will not have access. Defined roles to start will include owner, manager, cashier, and customer.

Authorization:

Authorization for the system will be based on user role first, not user account. By setting it up in this manner, it will be significantly easier to manage down the road as the business grows. If needed, additional roles can be added to contain differences between job functions. For example, if Vince adds a new position for an inventory specialist, they will have similar access as a cashier but will not need access to transaction data, so that will not be granted. However,

there will always be a chance that additional privileges will need to be assigned by the user. Authorization should always be assigned first by role to make the management and future updates for access easier.

To manage authorizations, access will be granted by schema where possible, meaning access would be the same for every table in the schema. However, where a role needs different access per table, the least access possible will be applied for the role to the schema, and then additional privileges will be given for the needed tables. For customers where they can access only their data, views can be utilized along with store procedures that input their customer ID so it can be ensured that they are only accessing their information and not that of another customer. A summary of the authorizations and privileges are shown below in Table 3.

Owner Permissions	SELECT	INSERT	UPDATE	DELETE	Constraints
Records.Inventory	X	X	X	X	
Records.Records	X	X	X	X	
Records.Conditions	X	X	X	X	
Records.RecordTypes	X	X	X	X	
Transactions.Transactions	X	X	X	X	
Transactions.ItemsToTransactions	X	X	X	X	
Transactions.Purchases	X	X	X	X	
Transactions.ItemsToPurchases	X	X	X	X	
Transactions.DiscountTypes	X	X	X	X	
Customers.Customers	X	X	X	X	
Customers.CustomerRequests	X	X	X	X	

Manager Permissions	SELECT	INSERT	UPDATE	DELETE	Constraints
Records.Inventory	X	X	X	X	
Records.Records	X	X	X	X	
Records.Conditions	X				
Records.RecordTypes	X				
Transactions.Transactions	X	X	X	X	
Transactions.ItemsToTransactions	X	X	X	X	
Transactions.Purchases	X				
Transactions.ItemsToPurchases	X				
Transactions.DiscountTypes	X				
Customers.Customers	X	X	X	X	
Customers.CustomerRequests	X	X	X		

Cashier Permissions	SELECT	INSERT	UPDATE	DELETE	Constraints
Records.Inventory	X		X		

Records.Records	X		X		
Records.Conditions	X				
Records.RecordTypes	X				
Transactions.Transactions	X	X	X		
Transactions.ItemsToTransactions	X	X	X		
Transactions.Purchases					
Transactions.ItemsToPurchases					
Transactions.DiscountTypes	X				
Customers.Customers	X	X	X		Can't select more than one at a time
Customers.CustomerRequests	X	X	X		

Customer Permissions	**SELECT**	**INSERT**	**UPDATE**	**DELETE**	**Constraints**
Records.Inventory	X				
Records.Records	X				
Records.Conditions	X				
Records.RecordTypes	X				
Transactions.Transactions	X				Only their own transactions
Transactions.ItemsToTransactions	X				Only their own transactions
Transactions.Purchases	X				Only their own transactions
Transactions.ItemsToPurchases	X				Only their own transactions
Transactions.DiscountTypes					
Customers.Customers	X				Only their own information
Customers.CustomerRequests	X	X	X	X	Only their own information

Table 3 These four tables describe the authorizations and privileges based on the current four user account types: owner, manager, cashier, and customer. The three schemas are separated and color-coded to match the entity-relationship diagram shown in Figure 4. Ideally, privileges would be assigned by schema but can also be granted by a table.

Policies and Procedures:

In general, each of the roles except for the owner will have the least privilege possible. This will ensure that data is not mistakenly updated or potentially stolen and misused. Employees and customers will be assigned unique usernames and passwords meeting minimum password requirements and will be required to reset them on a regular basis. By default, customers can only access their own information, and only managers can delete information from the database so it can be ensured that employees are not doing it by accident. Since Vince is the only employee purchasing albums, only he can enter and update that information in the database. As employees are terminated, their accounts will be revoked and removed from the system. Likewise, after a year of inactivity, customer accounts will become inactive and locked. Customers will need to reset their password and confirm their contact information in order to

gain access again to the system or will need to make a purchase so the store can mark their account as current again.

As mentioned, when researching the various database management systems, Vince will likely choose to utilize the cloud-based Amazon Aurora system through Amazon Web Services. By utilizing this software, he has the unique advantage of his database being continuously backed up to the cloud. This ensures that his data is always archived and backed up securely without needing to remember to do so on his end. Vince can schedule the backups during certain times of the day, such as overnight when the system is not being used frequently, so there isn't a slow down for customers and employees during peak times.

Additionally, for extra security and peace of mind, Vince should manually download his entire database to his computer or system on a weekly basis and save it to an external hard drive or server so that he always has a copy of everything in his possession as opposed to being solely in the cloud. In the event of data corruption, Vince can use the online backups first or one of his physical copies to recover as much data as possible. While it is unlikely that Vince will ever need to use these backups, it is better to have them and never use them than not have them and potentially lose all of his data. Likewise, if he would ever lose connection to Amazon Web Services, get locked out of his account, or not have access to the internet in general, he would still have mostly up-to-date information if he needed it.

Database Management Solution

Given that everything in Vince's store is currently done by memory or written down on paper instead of being digitalized, any database would significantly improve the owner and his employees' tasks. His current business operations are inefficient and do not protect the business should anything happen to Vince. It would be challenging for operations to continue without him

being in the store or being readily available. That being said, it is still best to carefully evaluate the needs of the business, design a robust database to meet the requirements, and choose a database management system to accompany it. As previously stated, Vince's database will be designed to contain three separate schemas with a total of eleven tables. The database management system used will be Amazon Web Services' RDS running the Amazon Aurora engine.

Of the options and configurations researched, this approach will lend itself to being the most manageable if Vince would need to dig into the tables at all, and it will be the most logical design for future updates. If able, it would be desirable to extend the complete package to include a customer-facing website and a user interface that will allow the employees to manage the customer's information, enter sales, and manage the day-to-day business functions. By doing so, Vince would be able to eliminate the need to directly interact with the database as it would be masked through the interface. Not only will this protect the data's integrity, but it will also help to create a complete, polished system that will significantly accelerate the growth of his business and customer base.

References

AWS. (2019). *Amazon Aurora*. Amazon Web Services, Inc. https://aws.amazon.com/rds/aurora/

Bennett, P. (n.d.). *What Is an Enterprise Data Model?* HubSpot; HubSpot, Inc. Retrieved November 24, 2021, from https://blog.hubspot.com/marketing/enterprise-data-model

Berecki, B. (2019, June 28). *6 Data Protection Laws for US Organizations*. Endpoint Protector; CoSoSys Ltd. https://www.endpointprotector.com/blog/6-data-protection-laws-for-us-organizations/

Conger, S. (2014). *Hands-on database: an introduction to database design and development*. Pearson.

De Mers, B. A. (2014, November 20). *On Ethical Issues Surrounding the Planning and Designing of Databases*. LinkedIn. https://www.linkedin.com/pulse/20141120200923-338627392-on-ethical-issues-surrounding-the-planning-and-designing-of-databases/

Edwards, W. (2021, October 22). *Managing data in a retail environment* (T. Latshaw, Interviewer) [Personal communication].

Gupta, R. (2021, May 6). *Using Database Schemas in SQL Server*. Quest Software Inc. https://www.quest.com/community/blogs/b/database-management/posts/using-database-schemas-in-sql-server

Hernandez, M. J. (2013). *Database design for mere mortals: a hands-on guide to relational database design*. Addison-Wesley, Cop.

IBM Cloud Education. (2019, August 6). *Relational Databases*. IBM. https://www.ibm.com/cloud/learn/relational-databases

IBM Cloud Education. (2021, June 14). *Business Rules*. IBM Cloud Education; IBM. https://www.ibm.com/cloud/learn/business-rules

Kendle, N. (2005, July 1). *The Enterprise Data Model*. The Data Administration Newsletter, LLC; DATAVERSITY Digital LLC. https://tdan.com/the-enterprise-data-model/5205

Klosowski, T. (2021, September 6). *The State of Consumer Data Privacy Laws in the US (And Why It Matters)*. Wirecutter, Inc.; The New York Times. https://www.nytimes.com/wirecutter/blog/state-of-privacy-laws-in-us/

Microsoft. (n.d.). *Database design basics*. Microsoft. https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5

Microsoft. (2019). *SQL Server 2019*. Microsoft. https://www.microsoft.com/en-us/sql-server/sql-server-2019

MySQL. (2000). *MySQL*. MySQL; Oracle Corporation. https://www.mysql.com/

Oracle. (n.d.). *Data Integrity*. Oracle Help Center. Retrieved October 24, 2021, from https://docs.oracle.com/en/database/oracle/oracle-database/21/cncpt/data-integrity.html

PostgreSQL. (2019). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. PostgreSQL; The PostgreSQL Global Development Group. https://www.postgresql.org/

Stewart, J. (2008). Summary of crow's foot notation [Online Image]. In *The Data Administration Newsletter*. https://tdan.com/crows-feet-are-best/7474

Taylor, D. (2018, December 24). *What is Data Modelling? Conceptual, Logical, & Physical Data Models*. Guru99. https://www.guru99.com/data-modelling-conceptual-logical.html

Wagner, B. (2019, October 15). *Are Stored Procedures Faster Than Stand-Alone Queries?* Bert Wagner. https://bertwagner.com/posts/are-stored-procedures-faster-than-stand-alone-queries/

Appendix A: Sample Reporting Queries

The following query could be executed to create a new database view called 'vw_transactions_over_registered_average' which could later be used to quickly find any registered user transaction where the price of the sale is greater than the average purchase price of registered users. A view would be beneficial in this scenario as it would mask all of the complex logic in the query as well as limit access to certain information.

```
CREATE VIEW Transactions.vw_transactions_over_registered_average AS
SELECT T.TransactionID
	,T.TransactionDateTime
	,T.Subtotal
	,T.TotalDiscounts
	,T.Tax
	,T.TotalPrice
	,C.CustomerID
	,CONCAT(C.FirstName, ' ', C.LastName) AS 'CustomerName'
FROM Transactions.Transactions T
JOIN Customers.Customers C
ON T.CustomerID = C.CustomerID
WHERE C.IsGuestPurchase = 0
AND T.TotalPrice >
	(SELECT CAST(AVG(T.TotalPrice) AS money)
	FROM Transactions.Transactions T
	JOIN Customers.Customers C
	ON T.CustomerID = C.CustomerID
	WHERE C.IsGuestPurchase = 0)
```

The following query could then be used to find the applicable sales. The start and end date parameters would be supplemented into the query from variables selected in the user interface being used.

```
DECLARE @StartDate datetime = '2013-05-12 00:00:00.000'
DECLARE @EndDate datetime = '2013-05-13 00:00:00.000'

SELECT *
FROM Transactions.vw_transactions_over_registered_average
WHERE TransactionDateTime >= @StartDate
AND TransactionDateTime < @EndDate
```

The following query could be used by Vince to report on all transactions – both purchases and sales. He could add any number of parameters such as a date, time, or customer restriction or join it with the respective linking tables to find specific items or conditions.

```
SELECT 'Purchase by Customer' AS 'Transaction Type'
	,T.TransactionID AS 'Transaction ID'
	,T.TransactionDateTime AS 'Transaction Date'
	,T.TotalPrice AS 'Transaction Amount'
	,T.CustomerID AS 'Customer ID'
	,CONCAT(C.FirstName, ' ', C.LastName) AS 'Customer Name'
```

```
FROM Transactions.Transactions T
LEFT JOIN Customers.Customers C
ON T.CustomerID = C.CustomerID
UNION ALL
SELECT 'Sale by Customer'
		,P.PurchaseID
		,P.PurchaseDateTime
		,P.TotalCost
		,P.CustomerID
		,CONCAT(C.FirstName, ' ', C.LastName)
FROM Transactions.Purchases P
LEFT JOIN Customers.Customers C
ON P.CustomerID = C.CustomerID
ORDER BY 'Transaction Date'
```

Appendix B: Sample Stored Procedures

The following query would create a new stored procedure in the ‘Customers’ schema called ‘stoAddRegisteredCustomer’ that would pass in variables to create a newly registered user in the system.

```
CREATE PROCEDURE Customers.stoAddRegisteredCustomer @CustomerID varchar(50)
	,@FirstName varchar(50)
	,@LastName varchar(50)
	,@Email varchar(50)
	,@Password char(128)
AS
BEGIN
	INSERT Customers.Customers (CustomerID
		,FirstName
		,LastName
		,Email
		,AveragePurchase
		,TotalPurchase
		,TotalSales
		,IsRegistered
		,Password
		,IsGuestPurchase)
	VALUES (@CustomerID
		,@FirstName
		,@LastName
		,@Email
		,0.0000
		,0.0000
		,0.0000
		,1
		,@Password
		,0)
	END
```

The following query will then execute the procedure and insert the new customer information. This could be executed from the cashier’s interface or when registering a new account online.

```
EXEC Customers.stoAddRegisteredCustomer @CustomerID = 'TL1001'
	,@FirstName = 'Tyler'
	,@LastName = 'Latshaw'
	,@Email = 'tyler.latshaw@snhu.edu'
	,@Password = 'O7YHITLDPS5Y9756W9LDQKFPEPFOE18MIFHUBSUCYU4HTTN5V9UA756OZAD2EQLS';
```

The following query would be used to create a new stored procedure in the ‘Transactions’ schema to create a new transaction or purchase made by a customer. This assumes that the customer’s ID is already known at the time of sale. Once the initial transaction is created, the ID will be returned and will be used in additional stored procedures to associate items with the transaction. The stock will later be updated in another stored procedure. Values such as the total

price and the tax would typically be calculated dynamically by the interface, not the database, so they will be shown as example numbers.

```
--Create stored procedure to add a new transaction

CREATE PROCEDURE Transactions.stoAddNewTransaction @TransactionID varchar(50)
	,@TransactionDateTime datetime
	,@CustomerID varchar(50)
	,@Subtotal money
	,@Tax money
	,@TotalCost money
	,@TotalPrice money
	,@TotalProfit money
	,@NumberOfItems tinyint
AS
BEGIN
	INSERT Transactions.Transactions (TransactionID
		,TransactionDateTime
		,CustomerID
		,Subtotal
		,Tax
		,TotalCost
		,TotalPrice
		,TotalProfit
		,NumberOfItems)
	VALUES (@TransactionID
		,@TransactionDateTime
		,@CustomerID
		,@Subtotal
		,@Tax
		,@TotalCost
		,@TotalPrice
		,@TotalProfit
		,@NumberOfItems)
END

--Insert new transaction

DECLARE @Current_Time datetime
SET @Current_Time = CURRENT_TIMESTAMP

EXEC Transactions.stoAddNewTransaction @TransactionID = '5925'
	,@TransactionDateTime = @Current_Time
	,@CustomerID = 'TL1001'
	,@Subtotal = '23.99'
	,@Tax = '1.44'
	,@TotalCost = '15.00'
	,@TotalPrice = '25.43'
	,@TotalProfit = '8.99'
	,@NumberOfItems = '2'

--Create stored procedure to add items to the transaction

CREATE PROCEDURE Transactions.stoAddItemsToTransaction @ItemsToTransactionID varchar(50)
	,@TransactionID varchar(50)
	,@InventoryID varchar(50)
	,@Cost money
	,@Price money
	,@Profit money
	,@OriginalPrice money
AS
BEGIN
	INSERT Transactions.ItemsToTransactions(ItemsToTransactionID
		,TransactionID
		,InventoryID
		,Cost
		,Price
```

```
                    ,Profit
                    ,OriginalPrice)
            VALUES (@ItemsToTransactionID
                    ,@TransactionID
                    ,@InventoryID
                    ,@Cost
                    ,@Price
                    ,@Profit
                    ,@OriginalPrice)
END

--Insert a new item to the transaction (would be executed as many times as the number of items)

EXEC Transactions.stoAddItemsToTransaction @ItemsToTransactionID = '15621'
                    ,@TransactionID = '5925'
                    ,@InventoryID = '10008'
                    ,@Cost = '0.50'
                    ,@Price = '4.20'
                    ,@Profit = '3.70'
                    ,@OriginalPrice = '4.20'

--Create stored procedure to update stock levels after purchase

CREATE PROCEDURE Records.stoUpdateStock @InventoryID varchar(50)
            ,@QuantityAvailable nchar(50)
AS
BEGIN
            UPDATE Records.Inventory
            SET QuantityAvailable = @QuantityAvailable
            WHERE InventoryID = @InventoryID
END

--Update stock levels after purchase

EXEC Records.stoUpdateStock @InventoryID = '10008'
            ,@QuantityAvailable = '7'
```

Appendix C: Database Creation Script

The following SQL script can be executed to create the entire database proposed for Vince's Vinyl. The script will create all of the necessary tables and seed the tables with the appropriate testing data that was provided in the case study from the interviews and operations shadowing by the designer.

```
USE [master] GO
CREATE DATABASE [VincesVinyl]
 CONTAINMENT = NONE
 ON PRIMARY
( NAME = N'VincesVinyl', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\VincesVinyl.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED,
FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'VincesVinyl_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\VincesVinyl_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB ,
FILEGROWTH = 65536KB ) GO
ALTER DATABASE [VincesVinyl] SET COMPATIBILITY_LEVEL = 140 GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [VincesVinyl].[dbo].[sp_fulltext_database] @action = 'enable'
end GO
ALTER DATABASE [VincesVinyl] SET ANSI_NULL_DEFAULT OFF GO
ALTER DATABASE [VincesVinyl] SET ANSI_NULLS OFF GO
ALTER DATABASE [VincesVinyl] SET ANSI_PADDING OFF GO
ALTER DATABASE [VincesVinyl] SET ANSI_WARNINGS OFF GO
ALTER DATABASE [VincesVinyl] SET ARITHABORT OFF GO
ALTER DATABASE [VincesVinyl] SET AUTO_CLOSE OFF GO
ALTER DATABASE [VincesVinyl] SET AUTO_SHRINK OFF GO
ALTER DATABASE [VincesVinyl] SET AUTO_UPDATE_STATISTICS ON GO
ALTER DATABASE [VincesVinyl] SET CURSOR_CLOSE_ON_COMMIT OFF GO
ALTER DATABASE [VincesVinyl] SET CURSOR_DEFAULT GLOBAL GO
ALTER DATABASE [VincesVinyl] SET CONCAT_NULL_YIELDS_NULL OFF GO
ALTER DATABASE [VincesVinyl] SET NUMERIC_ROUNDABORT OFF GO
ALTER DATABASE [VincesVinyl] SET QUOTED_IDENTIFIER OFF GO
ALTER DATABASE [VincesVinyl] SET RECURSIVE_TRIGGERS OFF GO
ALTER DATABASE [VincesVinyl] SET DISABLE_BROKER GO
ALTER DATABASE [VincesVinyl] SET AUTO_UPDATE_STATISTICS_ASYNC OFF GO
ALTER DATABASE [VincesVinyl] SET DATE_CORRELATION_OPTIMIZATION OFF GO
ALTER DATABASE [VincesVinyl] SET TRUSTWORTHY OFF GO
ALTER DATABASE [VincesVinyl] SET ALLOW_SNAPSHOT_ISOLATION OFF GO
ALTER DATABASE [VincesVinyl] SET PARAMETERIZATION SIMPLE GO
ALTER DATABASE [VincesVinyl] SET READ_COMMITTED_SNAPSHOT OFF GO
ALTER DATABASE [VincesVinyl] SET HONOR_BROKER_PRIORITY OFF GO
ALTER DATABASE [VincesVinyl] SET RECOVERY FULL GO
ALTER DATABASE [VincesVinyl] SET MULTI_USER GO
ALTER DATABASE [VincesVinyl] SET PAGE_VERIFY CHECKSUM GO
ALTER DATABASE [VincesVinyl] SET DB_CHAINING OFF GO
ALTER DATABASE [VincesVinyl] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF ) GO
ALTER DATABASE [VincesVinyl] SET TARGET_RECOVERY_TIME = 60 SECONDS GO
ALTER DATABASE [VincesVinyl] SET DELAYED_DURABILITY = DISABLED GO
EXEC sys.sp_db_vardecimal_storage_format N'VincesVinyl', N'ON' GO
ALTER DATABASE [VincesVinyl] SET QUERY_STORE = OFF GO
USE [VincesVinyl] GO
/****** Object: Schema [Customers] Script Date: 11/7/2021 3:31:08 PM ******/
CREATE SCHEMA [Customers] GO
/****** Object: Schema [Records] Script Date: 11/7/2021 3:31:08 PM ******/
CREATE SCHEMA [Records] GO
/****** Object: Schema [Transactions] Script Date: 11/7/2021 3:31:08 PM ******/
CREATE SCHEMA [Transactions] GO
/****** Object: Table [Customers].[CustomerRequests] Script Date: 11/7/2021 3:31:08 PM ******/
```

```
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Customers].[CustomerRequests](
	[RequestID] [varchar](50) NOT NULL,
	[CustomerID] [varchar](50) NOT NULL,
	[RecordID] [varchar](50) NOT NULL,
	[ConditionID] [tinyint] NOT NULL,
	[RecordTypeID] [tinyint] NOT NULL,
	[MaximumPrice] [money] NULL,
	[RequestNotes] [text] NULL,
	[DateFound] [date] NULL,
	[CustomerPurchased] [bit] NULL,
 CONSTRAINT [PK_CustomerRequests] PRIMARY KEY CLUSTERED
(
	[RequestID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO
/****** Object: Table [Customers].[Customers] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Customers].[Customers](
	[CustomerID] [varchar](50) NOT NULL,
	[FirstName] [varchar](50) NULL,
	[LastName] [varchar](50) NULL,
	[Email] [varchar](50) NULL,
	[PhoneNumber] [nchar](10) NULL,
	[StreetAddress] [varchar](50) NULL,
	[City] [varchar](50) NULL,
	[State] [varchar](50) NULL,
	[ZipCode] [nchar](5) NULL,
	[AveragePurchase] [money] NOT NULL,
	[TotalPurchase] [money] NOT NULL,
	[TotalSales] [money] NOT NULL,
	[IsRegistered] [bit] NOT NULL,
	[Password] [char](128) NULL,
	[IsGuestPurchase] [bit] NOT NULL,
 CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED
(
	[CustomerID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Records].[Conditions] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Records].[Conditions](
	[ConditionID] [tinyint] NOT NULL,
	[ConditionName] [varchar](50) NULL,
 CONSTRAINT [PK_Conditions] PRIMARY KEY CLUSTERED
(
	[ConditionID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Records].[Inventory] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Records].[Inventory](
	[InventoryID] [varchar](50) NOT NULL,
	[RecordID] [varchar](50) NOT NULL,
	[ConditionID] [tinyint] NOT NULL,
	[RecordTypeID] [tinyint] NOT NULL,
	[Cost] [money] NOT NULL,
	[Price] [money] NOT NULL,
	[QuantityAvailable] [nchar](10) NULL,
 CONSTRAINT [PK_Inventory] PRIMARY KEY CLUSTERED
(
	[InventoryID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Records].[Records] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Records].[Records](
	[RecordID] [varchar](50) NOT NULL,
	[AlbumTitle] [varchar](50) NOT NULL,
	[Artist] [varchar](50) NOT NULL,
	[Year] [smallint] NOT NULL,
	[AverageCost] [money] NULL,
	[AveragePrice] [money] NULL,
	[AverageProfit] [money] NULL,
	[Notes] [text] NULL,
 CONSTRAINT [PK_Records] PRIMARY KEY CLUSTERED
(
	[RecordID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO
/****** Object: Table [Records].[RecordTypes] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Records].[RecordTypes](
	[RecordTypeID] [tinyint] NOT NULL,
	[RecordTypeName] [varchar](50) NULL,
 CONSTRAINT [PK_RecordType] PRIMARY KEY CLUSTERED
(
	[RecordTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Transactions].[DiscountTypes] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Transactions].[DiscountTypes](
	[DiscountTypeID] [tinyint] NOT NULL,
	[DiscountName] [varchar](50) NOT NULL,
	[DiscountPercentage] [decimal](5, 0) NULL,
	[DiscountAmount] [money] NULL,
 CONSTRAINT [PK_DiscountTypes] PRIMARY KEY CLUSTERED
(
	[DiscountTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Transactions].[ItemsToPurchases] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Transactions].[ItemsToPurchases](
	[ItemsToPurchasesID] [varchar](50) NOT NULL,
	[PurchaseID] [varchar](50) NOT NULL,
	[InventoryID] [varchar](50) NOT NULL,
	[ConditionID] [tinyint] NOT NULL,
	[RecordTypeID] [tinyint] NOT NULL,
	[Cost] [money] NOT NULL,
 CONSTRAINT [PK_ItemsToPurchases] PRIMARY KEY CLUSTERED
(
	[ItemsToPurchasesID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Transactions].[ItemsToTransactions] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Transactions].[ItemsToTransactions](
	[ItemsToTransactionID] [varchar](50) NOT NULL,
	[TransactionID] [varchar](50) NOT NULL,
	[InventoryID] [varchar](50) NOT NULL,
```

```
	[Cost] [money] NOT NULL,
	[Price] [money] NOT NULL,
	[Profit] [money] NOT NULL,
	[OriginalPrice] [money] NOT NULL,
	[DiscountTypeID] [tinyint] NULL,
	[DiscountPercent] [decimal](5, 0) NULL,
	[DiscountAmount] [money] NULL,
 CONSTRAINT [PK_ItemsToTransactions] PRIMARY KEY CLUSTERED
(
	[ItemsToTransactionID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Transactions].[Purchases] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Transactions].[Purchases](
	[PurchaseID] [varchar](50) NOT NULL,
	[PurchaseDateTime] [datetime] NOT NULL,
	[CustomerID] [varchar](50) NOT NULL,
	[TotalCost] [money] NOT NULL,
	[NumberOfItems] [tinyint] NOT NULL,
 CONSTRAINT [PK_Purchases] PRIMARY KEY CLUSTERED
(
	[PurchaseID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
/****** Object: Table [Transactions].[Transactions] Script Date: 11/7/2021 3:31:08 PM ******/
SET ANSI_NULLS ON GO
SET QUOTED_IDENTIFIER ON GO
CREATE TABLE [Transactions].[Transactions](
	[TransactionID] [varchar](50) NOT NULL,
	[TransactionDateTime] [datetime] NOT NULL,
	[CustomerID] [varchar](50) NOT NULL,
	[Subtotal] [money] NOT NULL,
	[Tax] [money] NOT NULL,
	[TotalCost] [money] NOT NULL,
	[TotalPrice] [money] NOT NULL,
	[TotalProfit] [money] NOT NULL,
	[NumberOfItems] [tinyint] NOT NULL,
	[TotalDiscounts] [money] NULL,
 CONSTRAINT [PK_Transactions] PRIMARY KEY CLUSTERED
(
	[TransactionID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] GO
INSERT [Customers].[CustomerRequests] ([RequestID], [CustomerID], [RecordID], [ConditionID], [RecordTypeID],
[MaximumPrice], [RequestNotes], [DateFound], [CustomerPurchased]) VALUES (N'1', N'MC1001', N'10010', 2, 2, 25.0000, NULL,
CAST(N'2013-05-10' AS Date), 1)
INSERT [Customers].[CustomerRequests] ([RequestID], [CustomerID], [RecordID], [ConditionID], [RecordTypeID],
[MaximumPrice], [RequestNotes], [DateFound], [CustomerPurchased]) VALUES (N'2', N'LH1001', N'10012', 1, 2, 30.0000, NULL,
NULL, NULL) GO
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City],
[State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES
(N'BJ1001', N'Brad', N'Johnson', N'johnsonb@etown.edu', NULL, NULL, NULL, NULL, NULL, 5.4200, 5.4200, 0.0000, 0, NULL,
0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City],
[State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES
(N'GUEST', N'Guest', N'Guest', NULL, NULL, NULL, NULL, NULL, NULL, 17.8500, 3125.5000, 0.0000, 0, NULL, 1)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City],
[State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES
(N'JL1001', N'Jennifer', N'Louis', N'jennifer.margaret@hotmail.com', N'2065554545', N'1536 Birch Lane', N'Lancaster', N'PA',
N'17601', 10.2000, 101.7500, 61.5000, 1, N'C0051DF4E633CCADF7C592DBF98C8EAE14130038F89EB6A631D7021D234B8F04
', 0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City],
[State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES
(N'JL1002', N'John', N'Larson', N'jlars@gmail.com', N'2051164114', N'15 W Elm Street', N'Lancaster', N'PA', N'17602', 21.6000,
21.6000, 0.0000, 1, N'CFD968ADE49C10BA13B1C15A55C586831F2FECBA94ED3BBD662CEA5B1A63840E  ', 0)
```

```
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City], [State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES (N'JR1001', N'John', N'Raymond', N'jraymond@gmail.com', N'2065552352', N'123 Main Street', N'Lititz', N'PA', N'17543', 0.0000, 0.0000, 4.0000, 1, N'9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822CD15D6C15B0F00A08  ', 0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City], [State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES (N'LH1001', N'Laura', N'Hall', N'hall_family1@icloud.com', N'2065552080', N'54 Poplar Road', N'Lititz', N'PA', N'17543', 4.4500, 4.4500, 4.4500, 1, N'22A3EF9CED7185A902B8C85B1DCC22C52B7705215158433D0B225C65940E1537  ', 0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City], [State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES (N'MC1001', N'Maureen', N'Carlson', NULL, NULL, NULL, NULL, NULL, NULL, 25.1800, 25.1800, 0.0000, 0, NULL, 0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City], [State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES (N'MT1001', N'Marilyn', N'Taylor', N'music_gal@yahoo.com', N'2065550945', N'927 Elm Avenue', N'Elizabethtown', N'PA', N'17022', 4.6300, 32.7500, 4.7500, 1, N'231AA78C52A74A572E40F4A5DCDF24961945A8431E8061974151FFD9C78DA39C  ', 0)
INSERT [Customers].[Customers] ([CustomerID], [FirstName], [LastName], [Email], [PhoneNumber], [StreetAddress], [City], [State], [ZipCode], [AveragePurchase], [TotalPurchase], [TotalSales], [IsRegistered], [Password], [IsGuestPurchase]) VALUES (N'TS1001', N'Tabitha', N'Snyder', N'tabbysnyder@gmail.com', N'2011561561', N'73 Clymer Avenue', N'Reading', N'PA', N'19604', 17.7600, 17.7600, 175.0000, 0, NULL, 0) GO
INSERT [Records].[Conditions] ([ConditionID], [ConditionName]) VALUES (1, N'Mint')
INSERT [Records].[Conditions] ([ConditionID], [ConditionName]) VALUES (2, N'Good')
INSERT [Records].[Conditions] ([ConditionID], [ConditionName]) VALUES (3, N'Fair')
INSERT [Records].[Conditions] ([ConditionID], [ConditionName]) VALUES (4, N'Poor') GO
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10001', N'10005', 1, 1, 12.0000, 19.9500, N'0 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10002', N'10006', 2, 2, 2.8300, 5.9500, N'2 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10003', N'10001', 3, 2, 4.0000, 12.5000, N'0 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10004', N'10002', 2, 1, 4.7500, 11.9500, N'3 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10005', N'10003', 1, 3, 12.2500, 19.9500, N'1 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10006', N'10004', 2, 1, 4.4500, 10.9500, N'1 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10007', N'10007', 2, 2, 2.0000, 6.2500, N'2 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10008', N'10008', 4, 3, 0.5000, 4.2000, N'8 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10009', N'10010', 2, 2, 8.0000, 15.5000, N'0 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10010', N'10009', 4, 1, 0.0000, 5.0000, N'0 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10011', N'10011', 3, 2, 1.2500, 7.7500, N'0 ')
INSERT [Records].[Inventory] ([InventoryID], [RecordID], [ConditionID], [RecordTypeID], [Cost], [Price], [QuantityAvailable]) VALUES (N'10012', N'10010', 1, 3, 8.0000, 18.0000, N'1 ') GO
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10001', N'Rubber Soul', N'The Beatles', 1965, 4.0000, 16.0000, 14.0000, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10002', N'Led Zeppelin IV', N'Led Zeppelin', 1971, 4.8300, 16.2100, 11.3800, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10003', N'Gift of the flower to the Gardener', N'Donovan', 1989, 10.7300, 21.0000, 10.2700, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10004', N'Dark Side of the Moon', N'Pink Floyd', 1973, 5.2800, 12.1200, 6.8400, N'Customer favorite')
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10005', N'Blonde on Blonde', N'Bob Dylan', 1966, 3.3000, 19.9500, 16.6500, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10006', N'America', N'America', 1971, 0.0000, 5.9500, 5.9500, N'Limited release')
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10007', N'Blue', N'Joni Mitchell', 1968, 2.1700, 6.1800, 4.0100, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10008', N'Ballads', N'Joan Baez', 1985, 1.2000, 5.0000, 3.8000, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10009', N'Venus and Mars', N'Paul McCartney', 1979, 0.8500, 5.1700, 4.3200, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10010', N'The Crane Wife', N'The Decemberists', 2006, 9.0000, 15.5000, 6.5000, NULL)
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes]) VALUES (N'10011', N'Muddy Waters', N'Redman', 1996, 3.9800, 7.7800, 3.8000, NULL)
```

```
INSERT [Records].[Records] ([RecordID], [AlbumTitle], [Artist], [Year], [AverageCost], [AveragePrice], [AverageProfit], [Notes])
VALUES (N'10012', N'Mylo Xyloto', N'Coldplay', 2011, 0.0000, 0.0000, 0.0000, NULL) GO
INSERT [Records].[RecordTypes] ([RecordTypeID], [RecordTypeName]) VALUES (1, N'45')
INSERT [Records].[RecordTypes] ([RecordTypeID], [RecordTypeName]) VALUES (2, N'LP')
INSERT [Records].[RecordTypes] ([RecordTypeID], [RecordTypeName]) VALUES (3, N'76 RPM') GO
INSERT [Transactions].[DiscountTypes] ([DiscountTypeID], [DiscountName], [DiscountPercentage], [DiscountAmount]) VALUES
(1, N'10% Off', CAST(10 AS Decimal(5, 0)), NULL)
INSERT [Transactions].[DiscountTypes] ([DiscountTypeID], [DiscountName], [DiscountPercentage], [DiscountAmount]) VALUES
(2, N'20% Off', CAST(20 AS Decimal(5, 0)), NULL)
INSERT [Transactions].[DiscountTypes] ([DiscountTypeID], [DiscountName], [DiscountPercentage], [DiscountAmount]) VALUES
(3, N'$1 Off', NULL, 1.0000)
INSERT [Transactions].[DiscountTypes] ([DiscountTypeID], [DiscountName], [DiscountPercentage], [DiscountAmount]) VALUES
(4, N'$2 Off', NULL, 2.0000)
INSERT [Transactions].[DiscountTypes] ([DiscountTypeID], [DiscountName], [DiscountPercentage], [DiscountAmount]) VALUES
(5, N'50% Off', CAST(50 AS Decimal(5, 0)), NULL) GO
INSERT [Transactions].[ItemsToPurchases] ([ItemsToPurchasesID], [PurchaseID], [InventoryID], [ConditionID], [RecordTypeID],
[Cost]) VALUES (N'14551', N'2581', N'10003', 3, 2, 4.0000)
INSERT [Transactions].[ItemsToPurchases] ([ItemsToPurchasesID], [PurchaseID], [InventoryID], [ConditionID], [RecordTypeID],
[Cost]) VALUES (N'14552', N'2582', N'10004', 2, 1, 4.7500)
INSERT [Transactions].[ItemsToPurchases] ([ItemsToPurchasesID], [PurchaseID], [InventoryID], [ConditionID], [RecordTypeID],
[Cost]) VALUES (N'14553', N'2583', N'10005', 1, 3, 12.2500)
INSERT [Transactions].[ItemsToPurchases] ([ItemsToPurchasesID], [PurchaseID], [InventoryID], [ConditionID], [RecordTypeID],
[Cost]) VALUES (N'14554', N'2584', N'10006', 2, 1, 4.4500) GO
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15614', N'5161', N'10001', 12.0000, 19.9500,
7.9500, 19.9500, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15615', N'5162', N'10002', 2.8300, 5.9500,
3.1200, 5.9500, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15616', N'5162', N'10007', 2.0000, 6.2500,
4.2500, 6.2500, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15617', N'5162', N'10008', 0.5000, 4.2000,
3.7000, 4.2000, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15618', N'5163', N'10010', 0.0000, 5.0000,
5.0000, 5.0000, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15619', N'5164', N'10009', 8.0000, 15.5000,
7.5000, 15.5000, NULL, NULL, NULL)
INSERT [Transactions].[ItemsToTransactions] ([ItemsToTransactionID], [TransactionID], [InventoryID], [Cost], [Price], [Profit],
[OriginalPrice], [DiscountTypeID], [DiscountPercent], [DiscountAmount]) VALUES (N'15620', N'5164', N'10011', 1.2500, 7.7500,
6.5000, 7.7500, NULL, NULL, NULL) GO
INSERT [Transactions].[Purchases] ([PurchaseID], [PurchaseDateTime], [CustomerID], [TotalCost], [NumberOfItems]) VALUES
(N'2581', CAST(N'2013-05-12T00:00:00.000' AS DateTime), N'JR1001', 4.0000, 1)
INSERT [Transactions].[Purchases] ([PurchaseID], [PurchaseDateTime], [CustomerID], [TotalCost], [NumberOfItems]) VALUES
(N'2582', CAST(N'2013-05-12T00:00:00.000' AS DateTime), N'MT1001', 4.7500, 1)
INSERT [Transactions].[Purchases] ([PurchaseID], [PurchaseDateTime], [CustomerID], [TotalCost], [NumberOfItems]) VALUES
(N'2583', CAST(N'2013-05-12T00:00:00.000' AS DateTime), N'JL1001', 12.2500, 1)
INSERT [Transactions].[Purchases] ([PurchaseID], [PurchaseDateTime], [CustomerID], [TotalCost], [NumberOfItems]) VALUES
(N'2584', CAST(N'2013-05-12T00:00:00.000' AS DateTime), N'LH1001', 4.4500, 1) GO
INSERT [Transactions].[Transactions] ([TransactionID], [TransactionDateTime], [CustomerID], [Subtotal], [Tax], [TotalCost],
[TotalPrice], [TotalProfit], [NumberOfItems], [TotalDiscounts]) VALUES (N'5161', CAST(N'2013-05-12T09:24:00.000' AS
DateTime), N'JL1002', 19.9500, 1.6500, 12.0000, 21.6000, 7.9500, 1, 0.0000)
INSERT [Transactions].[Transactions] ([TransactionID], [TransactionDateTime], [CustomerID], [Subtotal], [Tax], [TotalCost],
[TotalPrice], [TotalProfit], [NumberOfItems], [TotalDiscounts]) VALUES (N'5162', CAST(N'2013-05-12T10:28:00.000' AS
DateTime), N'TS1001', 16.4000, 1.3600, 3.3700, 17.7600, 13.0300, 3, 0.0000)
INSERT [Transactions].[Transactions] ([TransactionID], [TransactionDateTime], [CustomerID], [Subtotal], [Tax], [TotalCost],
[TotalPrice], [TotalProfit], [NumberOfItems], [TotalDiscounts]) VALUES (N'5163', CAST(N'2013-05-12T11:16:00.000' AS
DateTime), N'BJ1001', 5.0000, 0.4200, 0.0000, 5.4200, 5.0000, 1, 0.0000)
INSERT [Transactions].[Transactions] ([TransactionID], [TransactionDateTime], [CustomerID], [Subtotal], [Tax], [TotalCost],
[TotalPrice], [TotalProfit], [NumberOfItems], [TotalDiscounts]) VALUES (N'5164', CAST(N'2013-05-12T12:26:00.000' AS
DateTime), N'MC1001', 29.2500, 1.9298, 9.2500, 25.1800, 20.0000, 2, 0.0000) GO
ALTER TABLE [Customers].[CustomerRequests] WITH CHECK ADD CONSTRAINT [FK_CustomerRequests_Conditions]
FOREIGN KEY([ConditionID])
REFERENCES [Records].[Conditions] ([ConditionID])
ON UPDATE CASCADE GO
ALTER TABLE [Customers].[CustomerRequests] CHECK CONSTRAINT [FK_CustomerRequests_Conditions] GO
```

```
ALTER TABLE [Customers].[CustomerRequests] WITH CHECK ADD CONSTRAINT [FK_CustomerRequests_Customers]
FOREIGN KEY([CustomerID])
REFERENCES [Customers].[Customers] ([CustomerID])
ON UPDATE CASCADE GO
ALTER TABLE [Customers].[CustomerRequests] CHECK CONSTRAINT [FK_CustomerRequests_Customers] GO
ALTER TABLE [Customers].[CustomerRequests] WITH CHECK ADD CONSTRAINT [FK_CustomerRequests_Records]
FOREIGN KEY([RecordID])
REFERENCES [Records].[Records] ([RecordID])
ON UPDATE CASCADE GO
ALTER TABLE [Customers].[CustomerRequests] CHECK CONSTRAINT [FK_CustomerRequests_Records] GO
ALTER TABLE [Customers].[CustomerRequests] WITH CHECK ADD CONSTRAINT [FK_CustomerRequests_RecordTypes]
FOREIGN KEY([RecordTypeID])
REFERENCES [Records].[RecordTypes] ([RecordTypeID])
ON UPDATE CASCADE GO
ALTER TABLE [Customers].[CustomerRequests] CHECK CONSTRAINT [FK_CustomerRequests_RecordTypes] GO
ALTER TABLE [Records].[Inventory] WITH CHECK ADD CONSTRAINT [FK_Inventory_Conditions] FOREIGN
KEY([ConditionID])
REFERENCES [Records].[Conditions] ([ConditionID])
ON UPDATE CASCADE GO
ALTER TABLE [Records].[Inventory] CHECK CONSTRAINT [FK_Inventory_Conditions] GO
ALTER TABLE [Records].[Inventory] WITH CHECK ADD CONSTRAINT [FK_Inventory_Records] FOREIGN KEY([RecordID])
REFERENCES [Records].[Records] ([RecordID])
ON UPDATE CASCADE GO
ALTER TABLE [Records].[Inventory] CHECK CONSTRAINT [FK_Inventory_Records] GO
ALTER TABLE [Records].[Inventory] WITH CHECK ADD CONSTRAINT [FK_Inventory_RecordTypes] FOREIGN
KEY([RecordTypeID])
REFERENCES [Records].[RecordTypes] ([RecordTypeID])
ON UPDATE CASCADE GO
ALTER TABLE [Records].[Inventory] CHECK CONSTRAINT [FK_Inventory_RecordTypes] GO
ALTER TABLE [Transactions].[ItemsToPurchases] WITH CHECK ADD CONSTRAINT [FK_ItemsToPurchases_Conditions]
FOREIGN KEY([ConditionID])
REFERENCES [Records].[Conditions] ([ConditionID]) GO
ALTER TABLE [Transactions].[ItemsToPurchases] CHECK CONSTRAINT [FK_ItemsToPurchases_Conditions] GO
ALTER TABLE [Transactions].[ItemsToPurchases] WITH CHECK ADD CONSTRAINT [FK_ItemsToPurchases_Inventory]
FOREIGN KEY([InventoryID])
REFERENCES [Records].[Inventory] ([InventoryID]) GO
ALTER TABLE [Transactions].[ItemsToPurchases] CHECK CONSTRAINT [FK_ItemsToPurchases_Inventory] GO
ALTER TABLE [Transactions].[ItemsToPurchases] WITH CHECK ADD CONSTRAINT [FK_ItemsToPurchases_Purchases]
FOREIGN KEY([PurchaseID])
REFERENCES [Transactions].[Purchases] ([PurchaseID]) GO
ALTER TABLE [Transactions].[ItemsToPurchases] CHECK CONSTRAINT [FK_ItemsToPurchases_Purchases] GO
ALTER TABLE [Transactions].[ItemsToPurchases] WITH CHECK ADD CONSTRAINT [FK_ItemsToPurchases_RecordTypes]
FOREIGN KEY([RecordTypeID])
REFERENCES [Records].[RecordTypes] ([RecordTypeID]) GO
ALTER TABLE [Transactions].[ItemsToPurchases] CHECK CONSTRAINT [FK_ItemsToPurchases_RecordTypes] GO
ALTER TABLE [Transactions].[ItemsToTransactions] WITH CHECK ADD CONSTRAINT
[FK_ItemsToTransactions_DiscountTypes] FOREIGN KEY([DiscountTypeID])
REFERENCES [Transactions].[DiscountTypes] ([DiscountTypeID]) GO
ALTER TABLE [Transactions].[ItemsToTransactions] CHECK CONSTRAINT [FK_ItemsToTransactions_DiscountTypes] GO
ALTER TABLE [Transactions].[ItemsToTransactions] WITH CHECK ADD CONSTRAINT [FK_ItemsToTransactions_Inventory]
FOREIGN KEY([InventoryID])
REFERENCES [Records].[Inventory] ([InventoryID])
ON UPDATE CASCADE GO
ALTER TABLE [Transactions].[ItemsToTransactions] CHECK CONSTRAINT [FK_ItemsToTransactions_Inventory] GO
ALTER TABLE [Transactions].[ItemsToTransactions] WITH CHECK ADD CONSTRAINT
[FK_ItemsToTransactions_Transactions] FOREIGN KEY([TransactionID])
REFERENCES [Transactions].[Transactions] ([TransactionID])
ON UPDATE CASCADE GO
ALTER TABLE [Transactions].[ItemsToTransactions] CHECK CONSTRAINT [FK_ItemsToTransactions_Transactions] GO
ALTER TABLE [Transactions].[Purchases] WITH CHECK ADD CONSTRAINT [FK_Purchases_Customers] FOREIGN
KEY([CustomerID])
REFERENCES [Customers].[Customers] ([CustomerID])
ON UPDATE CASCADE GO
ALTER TABLE [Transactions].[Purchases] CHECK CONSTRAINT [FK_Purchases_Customers] GO
ALTER TABLE [Transactions].[Transactions] WITH CHECK ADD CONSTRAINT [FK_Transactions_Customers] FOREIGN
KEY([CustomerID])
REFERENCES [Customers].[Customers] ([CustomerID])
ON UPDATE CASCADE GO
ALTER TABLE [Transactions].[Transactions] CHECK CONSTRAINT [FK_Transactions_Customers] GO
```

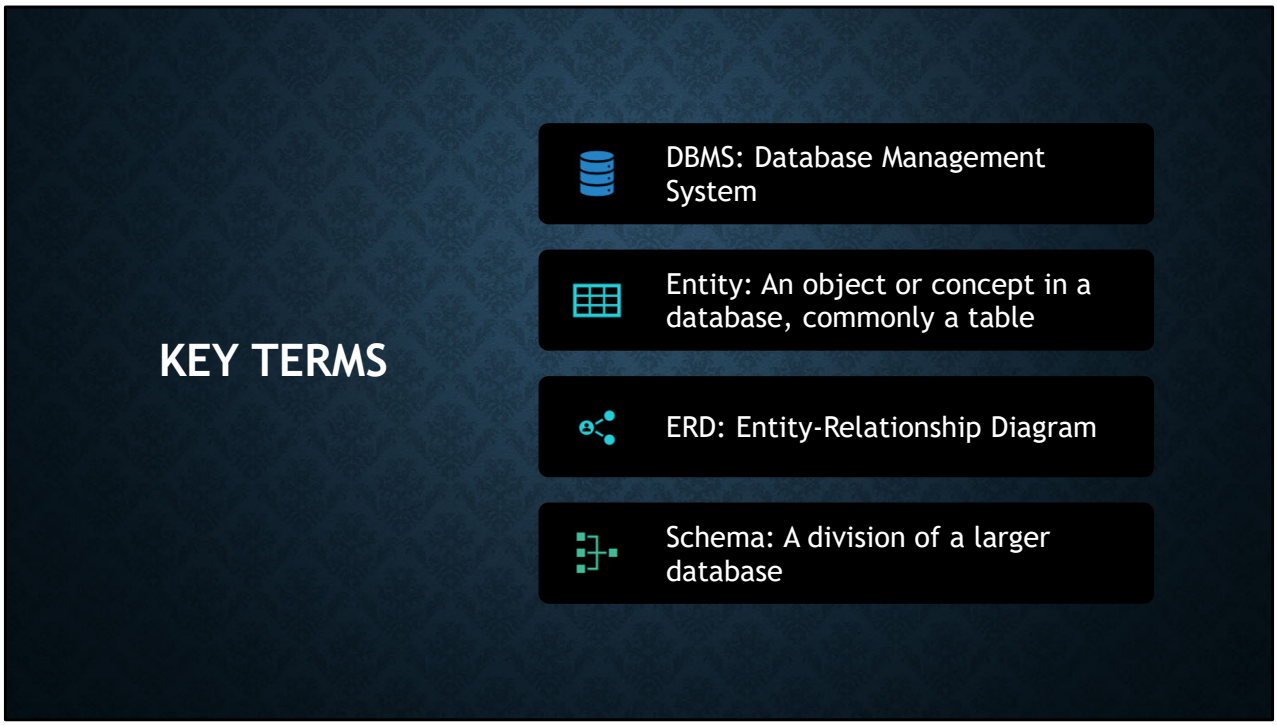
```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the individual request' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'RequestID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the customer making the request' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'CustomerID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the requested record' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'RecordID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the requested condition' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'ConditionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the requested record type' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'RecordTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The maximum price the customer is willing to pay' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'MaximumPrice' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Notes about the request' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'RequestNotes' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The date the record was found' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'DateFound' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Did the customer purchase the requested record after it
was found' , @level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests',
@level2type=N'COLUMN',@level2name=N'CustomerPurchased' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The requests for new albums by customers' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'CustomerRequests' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The unique ID of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'CustomerID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The first name of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'FirstName' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The last name of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'LastName' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The email address of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'Email' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The phone number of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'PhoneNumber' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The street address of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'StreetAddress' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The city of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'City' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The state of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'State' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The zipcode of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'ZipCode' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The average purchase amount of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'AveragePurchase' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total value of purchases by the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'TotalPurchase' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total value of sales from the customer to the
business' , @level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'TotalSales' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Is the customer registered or not' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'IsRegistered' GO
```

```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The encrypted password of the customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'Password' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Is the purchase by a guest or a known customer' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers',
@level2type=N'COLUMN',@level2name=N'IsGuestPurchase' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The customers that buy or sell with the business' ,
@level0type=N'SCHEMA',@level0name=N'Customers', @level1type=N'TABLE',@level1name=N'Customers' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The unique ID of the condition' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Conditions',
@level2type=N'COLUMN',@level2name=N'ConditionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The name of the condition' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Conditions',
@level2type=N'COLUMN',@level2name=N'ConditionName' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'A lookup table of conditions' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Conditions' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The unique ID number of each individual record' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'InventoryID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the record for the individual item' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'RecordID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The condition of the inventory item' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'ConditionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The type or format of record' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'RecordTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The cost to the business for the item' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'Cost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The price to the customer for the item' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'Price' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The number of identical records available' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Inventory',
@level2type=N'COLUMN',@level2name=N'QuantityAvailable' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the record regardless of condition or format' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'RecordID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The title of the album' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'AlbumTitle' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The artist of the album' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'Artist' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The year the album was released' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'Year' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The average cost of the record to the business' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'AverageCost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The average price the record is sold for' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'AveragePrice' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The average profit made on the record' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'AverageProfit' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Notes about the album' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records',
@level2type=N'COLUMN',@level2name=N'Notes' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The individual records that are sold regardless of
condition or format' , @level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'Records' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the type of record' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'RecordTypes',
@level2type=N'COLUMN',@level2name=N'RecordTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The name of the record type format' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'RecordTypes',
@level2type=N'COLUMN',@level2name=N'RecordTypeName' GO
```

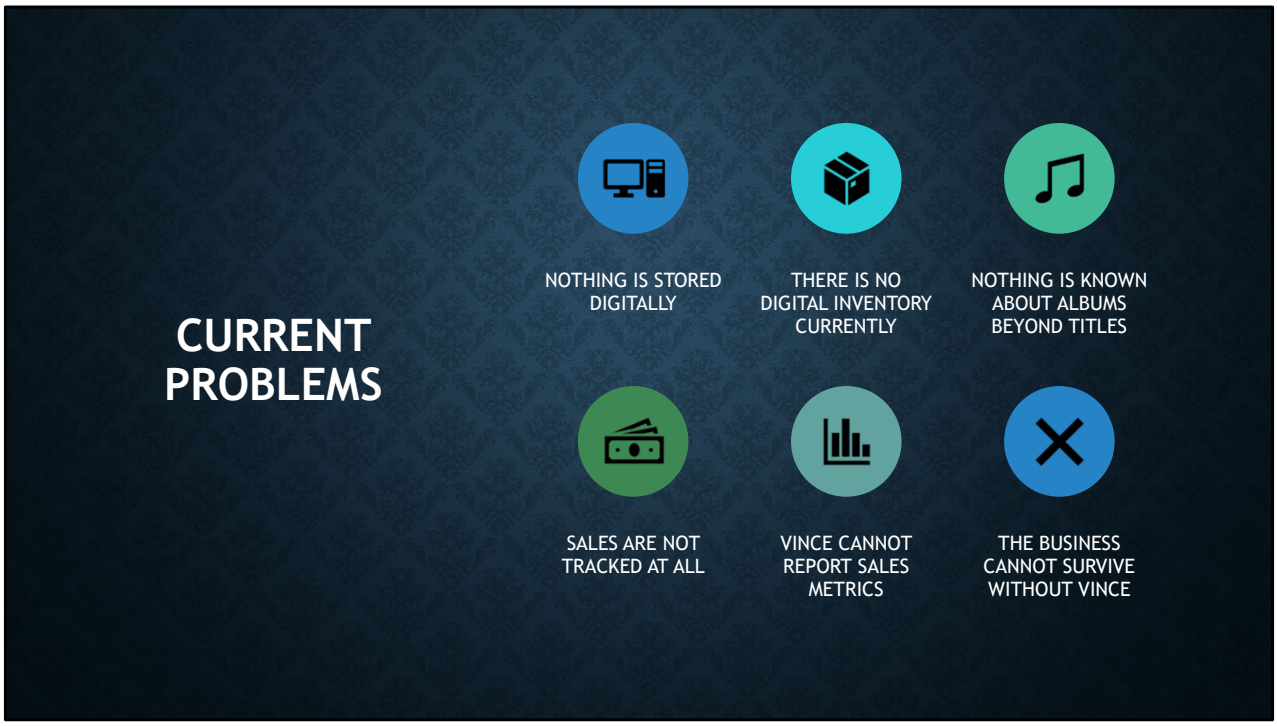
```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The type or formats of records sold' ,
@level0type=N'SCHEMA',@level0name=N'Records', @level1type=N'TABLE',@level1name=N'RecordTypes' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the discount' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'DiscountTypes',
@level2type=N'COLUMN',@level2name=N'DiscountTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The name of the discount' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'DiscountTypes',
@level2type=N'COLUMN',@level2name=N'DiscountName' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The percentage discounted off the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'DiscountTypes',
@level2type=N'COLUMN',@level2name=N'DiscountPercentage' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The flat amount discounted off the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'DiscountTypes',
@level2type=N'COLUMN',@level2name=N'DiscountAmount' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'A lookup table of discount types given to customers' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'DiscountTypes' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the item associated with a purchase by the
business' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'ItemsToPurchasesID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the purchase by the business' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'PurchaseID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the inventory item purchased' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'InventoryID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the condition of the item purhcased' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'ConditionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the type of record being purchased' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'RecordTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The cost of the individual item being purchased' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToPurchases',
@level2type=N'COLUMN',@level2name=N'Cost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The individual items associated with a purchase from a
customer by the business' , @level0type=N'SCHEMA',@level0name=N'Transactions',
@level1type=N'TABLE',@level1name=N'ItemsToPurchases' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the individual inventory item associated with a
transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions',
@level1type=N'TABLE',@level1name=N'ItemsToTransactions', @level2type=N'COLUMN',@level2name=N'ItemsToTransactionID'
GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the transaction' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'TransactionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the inventory item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'InventoryID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The cost of the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'Cost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The price of the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'Price' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The profit made on the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'Profit' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The price of the item before discounts' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'OriginalPrice' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the discount given on the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'DiscountTypeID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The percentage discount off the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'DiscountPercent' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The flat discount off the item' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions',
@level2type=N'COLUMN',@level2name=N'DiscountAmount' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The items associated with each transaction' ,
@level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'ItemsToTransactions' GO
```

```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The unique ID of a purhcase by the business from a customer' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases', @level2type=N'COLUMN',@level2name=N'PurchaseID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The date and time of the purchase' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases', @level2type=N'COLUMN',@level2name=N'PurchaseDateTime' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the customer that the records were purchased from' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases', @level2type=N'COLUMN',@level2name=N'CustomerID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total cost of the records being purchased at that time' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases', @level2type=N'COLUMN',@level2name=N'TotalCost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total number of items being purchased' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases', @level2type=N'COLUMN',@level2name=N'NumberOfItems' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The individual purchases of new records by the busines from a customer' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Purchases' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The unique ID of the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TransactionID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The date and time of the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TransactionDateTime' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The ID of the customer making the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'CustomerID' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The subtotal of the purchase before tax and discounts' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'Subtotal' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The amount of sales tax added to the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'Tax' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total cost of the items to the business' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TotalCost' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total price to the customer for the transaction including tax and discounts' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TotalPrice' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total profit made on the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TotalProfit' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total number of line items on the transaction' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'NumberOfItems' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The total amount of discounts on the purchase' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions', @level2type=N'COLUMN',@level2name=N'TotalDiscounts' GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'The individual transactions by customers' , @level0type=N'SCHEMA',@level0name=N'Transactions', @level1type=N'TABLE',@level1name=N'Transactions' GO
USE [master] GO
ALTER DATABASE [VincesVinyl] SET READ_WRITE GO
```

Hello, my name is Tyler Latshaw. I am the database designer working with Vince on this project. The overall goal of this project is to create a complete database management system implementation for Vince's business. The system and the database will store everything needed to run the day-to-day operations, including the inventory, the customers, and all of the transactions.



Before we get started, there are a few key terms that I want to highlight as they may not be familiar with everyone but will be referenced throughout this proposal. First and foremost, you will likely hear the term DBMS or database management system. That is the entire application or interface that allows a user to interact with a database. Entities are objects within a database, usually a table. In a retail database like this one, that would be a customer, a transaction, or similar. They are the concepts that we store data on. Think of them as individual excel files. An ERD or entity-relationship diagram is the visual layout of the database and how data is connected. You will see an example of one later. Lastly, schemas are how we will group entities together in the database. For example, anything related to transactions or purchases we could group together.



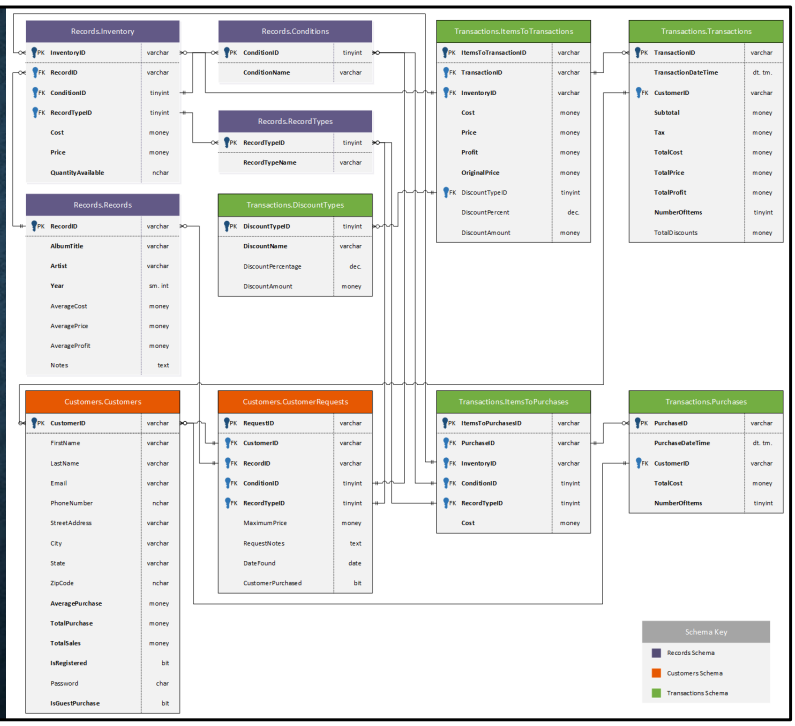
It's important to discuss some of the current problems that are being seen, as some of them are pretty impactful to the business. Right now, nothing is digitalized. Everything is either in Vince's memory or occasionally written down on paper. For obvious reasons, this is not ideal as the business would not be able to survive without Vince being there. There is no official record of the entire inventory, so no one would be able to search for a specific title. By extension, there's no record of any information beyond the titles, so nothing about the record year or publisher. Additionally, nothing is tracked about sales so Vince has no way to look at sales metrics to see how his store is performing.

DATABASE DESIGN

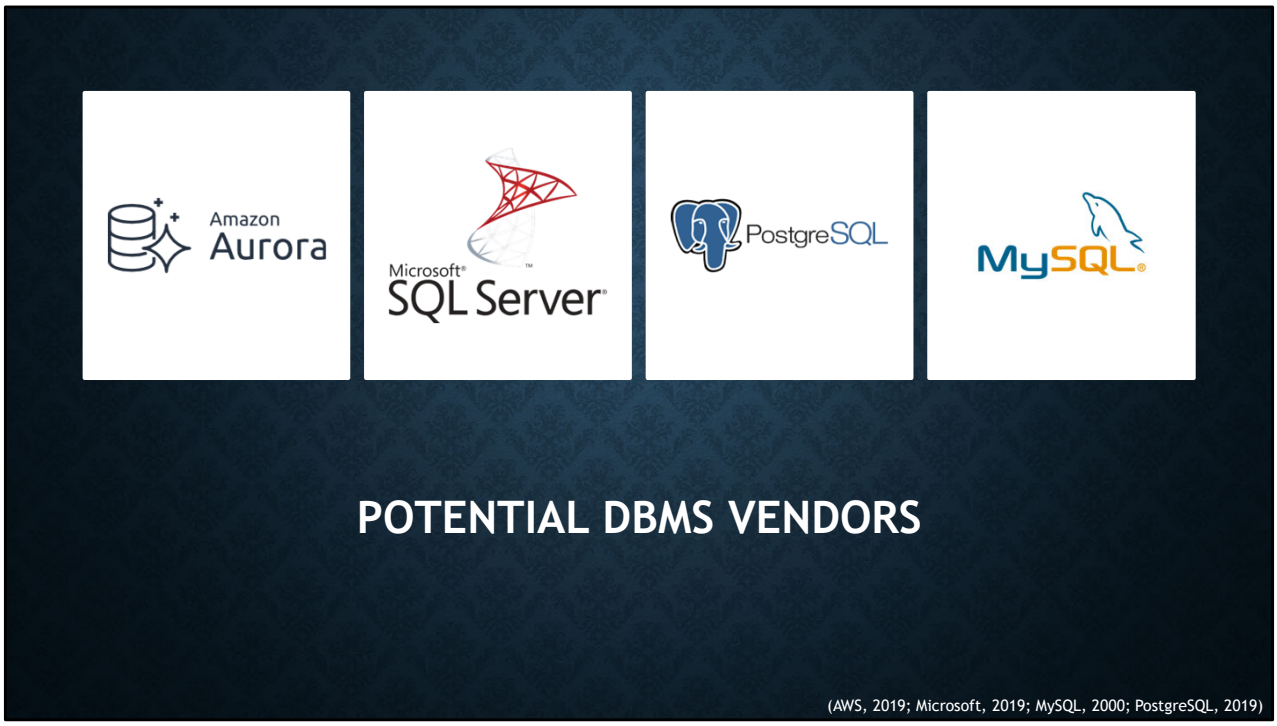
- Database will be split into three schemas
- Schemas allow for:
  - Logical separation of tables
  - Increased security
  - Easier management of data
- Database includes 11 tables:
  - 8 standards tables
  - 3 linking tables
  - 3 lookup tables

For the design of the database, we continually refined the design into what it is now. We will be taking the approach of breaking it up into three schemas for the records, the customers, and the transactions. Approaching it in schemas will allow for logical separation of the data while increasing security and the management of data. The proposed design includes eight tables as well as three linking and lookup tables, respectively.

PHYSICAL DESIGN



Here is the entity-relationship diagram or ERD that was created for the proposed database design. You can see all 11 of the tables listed out as well as the columns we will be stored in each table. These are referred to as attributes. For example, for customers, we will store their name, email, phone number, address, and relevant sales information. You'll also notice that the schemas are color-coded. The records-related tables are colored purple, customers are orange, and transactions are green.



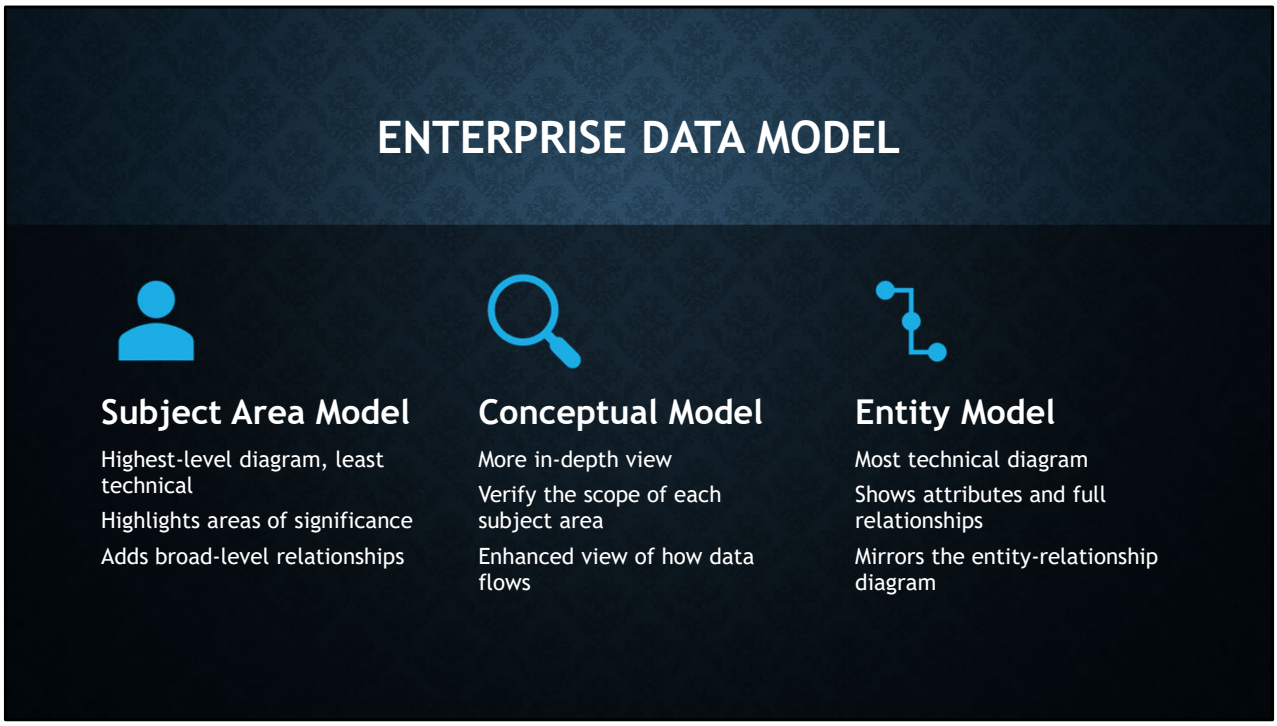
Now the part that you will most likely be interested in is how we are selecting a DBMS. This application will be the actual backbone of the entire system and will be how different configuration options are set up. Technically you could also edit the data through this, but the proposed system would just utilize a custom user interface instead. I do want to acknowledge that we did look into multiple DBMS vendors. These are the top four - Amazon Aurora through Amazon Web Services, Microsoft SQL Server, PostgreSQL, and MySQL.

DBMS CONSIDERATIONS

- Considerations and factors include:
  - Licensing type
  - Implementation format
  - Price
  - Advantages and disadvantages over competitors
  - Possible integrations
  - Other current customers

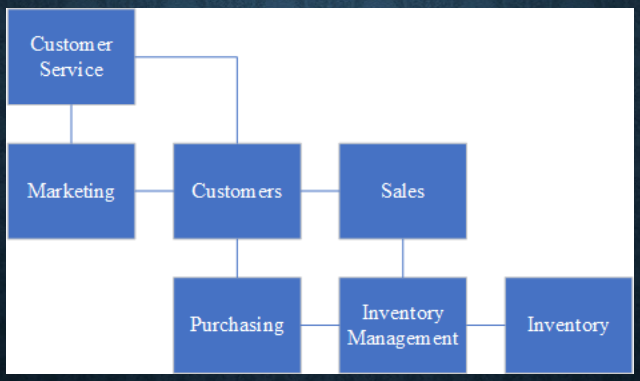
When comparing the four potential DBMS vendors, I compared a few different factors, including the license type, the format it was in, either in the cloud or stored on-premises in the store, the overall price, key differences between the vendors that would set them apart, as well as other integrations that could benefit Vince and what other companies are currently using the products. While we could go into other factors, these typically should be encompassing enough to give a good distinction between the vendors.

The formal recommendation for this proposal is Amazon's Aurora platform. The main advantage here is that it is completely scalable as the company grows and you will only need to pay for what you use. Instead of having to buy an expensive package now but not reap the full benefits until the company expands, you can use it to grow gradually. Additionally, this was the only option that was cloud-based so there won't be a need for new hardware. It is also fully managed by AWS and includes a number of integrations and performance metrics that Vince could tap into. I want to highlight the integrations as that is a key feature. Being able to use the rest of the AWS suite would allow us to quickly stand up a website or interface that could be used to handle all of the transactions and allow customers to shop the inventory from their home or phone.



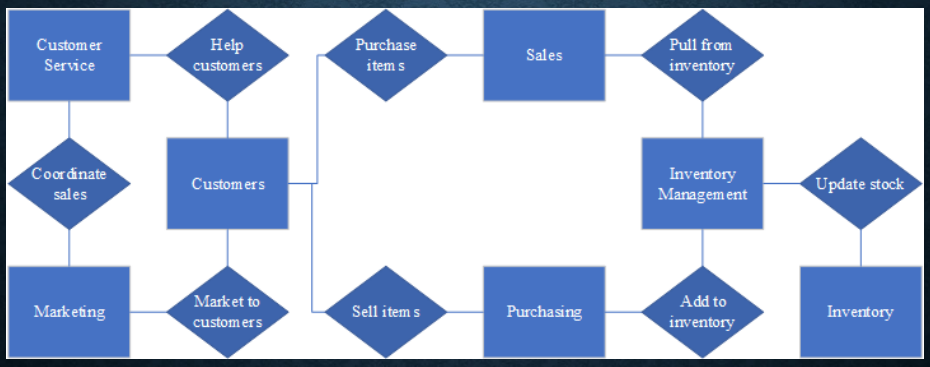
It is also important to take a look at specific teams or departments within the business to ensure that their needs are met and that they aren't negatively impacted by the new design. To do this, we can employ an enterprise data model. The model includes three specific models or diagrams including the subject area model, the conceptual model, and the entity model. The subject area model is the least technical and highest-level, showing broad relationships, whereas the entity model is very specific and will mirror the ERD. The conceptual model will be in between and combine elements of both. For these, we will focus specifically on the inventory management team.

SUBJECT AREA MODEL



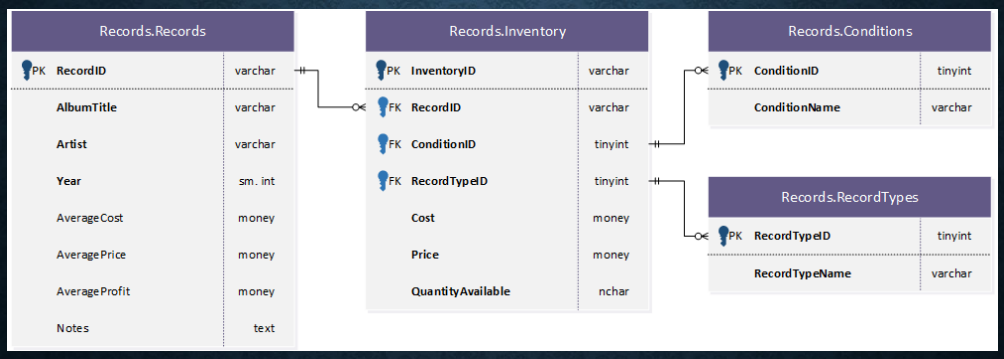
Here is an example of the subject area model. It shows the general subject areas that are of concern regarding inventory management. It shows how the data flows through the system.

CONCEPTUAL MODEL



This conceptual model expands upon the subject area model that I just showed you. The only difference here is that we start to include additional actions such as updating stock. Customer service won't directly update the stock as it needs to pass through the inventory management team per business rules. This diagram starts to show those, so as a designer, we know how to design everything properly.

ENTITY MODEL



The entity model is the last component of the enterprise data model. You'll notice that it very closely mirrors the ERD that you saw earlier. The main difference here is that this is just a subset of information, only surrounding the inventory.

Lastly, it is important to talk about database security when we are looking at this proposal. In terms of legal or ethical requirements, there aren't any overarching laws that need to be followed, but states do have different data protection requirements. Most recently, you may have heard about the California Consumer Protection Act or CCPA. Depending on if the store plans to do business in California or other states, those laws will need to be followed. While not a legal requirement, ethically, the database should follow the PCI DSS or payment card industry standards since the store accepts credit cards. Further, it is just ethical to ensure the data's security and integrity where possible.

SECURITY PLAN: AUTHENTICATION

- All users will have their own account
- Employees are will have an assigned login
- Accounts cannot be shared between employees
- User role types will be created
- Customers login through the internet once registered

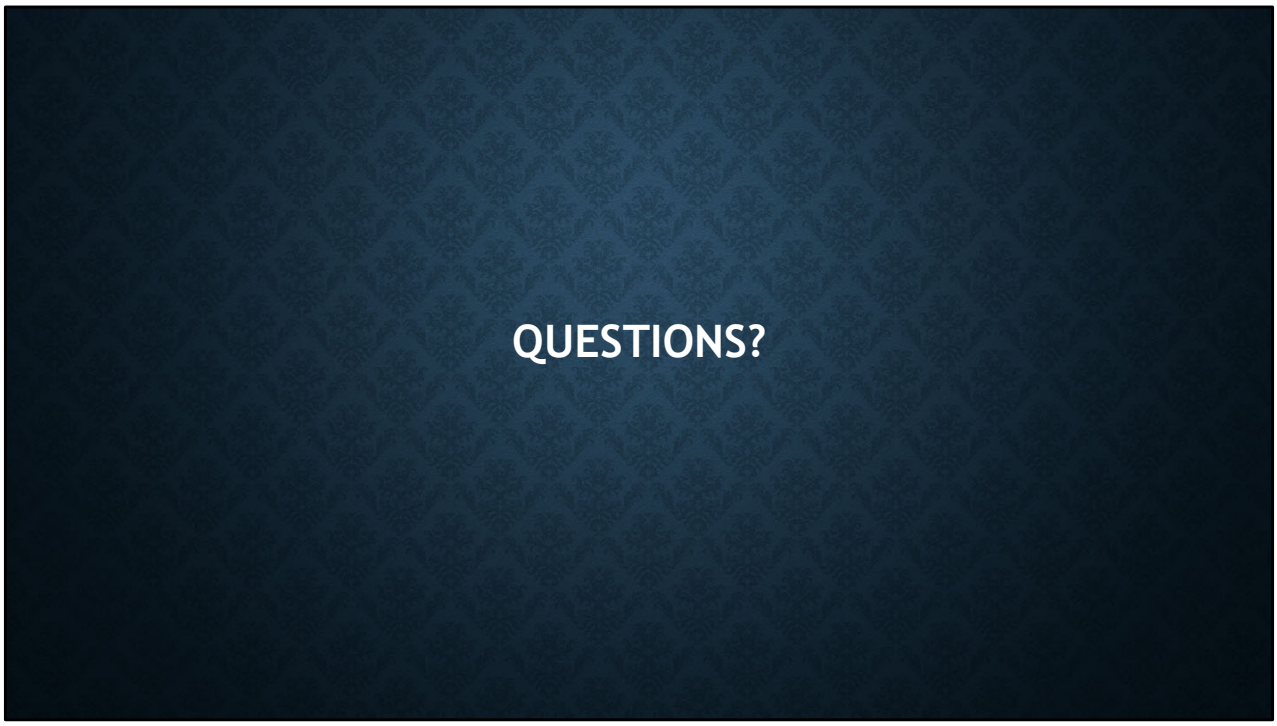
In terms of authenticating access, all users will have their own accounts - both employees and customers. Employees should be forbidden from sharing the login information that is assigned to them. Their access will be given by roles so we will likely have an owner role, a manager role, a cashier role, and a customer role. Access will be given based on the role. Customers will also be given access once they register with the company or make a purchase and provide their phone number.

Authorization within the system will be based on roles, as I mentioned. This will allow for easier permission management as Vince can grant access to data based on a role instead of needing to assign it per employee. We will follow the least privilege principle, meaning you are only given access to what you need, never more. Also, it goes without saying that customers will only see their information; they won't be able to see other customers' accounts.

SECURITY PLAN: POLICIES

- Only managers can delete information
- Terminated employees have their access revoked
- Inactive customers become locked out
- All users routinely need to reset passwords
- Automated processes will backup all data
- Backups will be cloud-based and physical

The last part of the proposed security plan is the list of policies that are going to be implemented from a business side. First and foremost, only managers can delete information from the database, such as completed transactions. This will help ensure data isn't accidentally deleted. Further, employees will lose access when they leave the store or are fired, and customers will become locked out when they don't access their accounts for a year. Along with this, users will need to routinely reset their passwords based on industry standards, usually every three to six months. The last part of this is related to the backups of the database. With Amazon Aurora, we can configure the DBMS to routinely create automated backups overnight, so the data is always there, even if something breaks on Amazon's side. Additionally, it would be up to an employee, but it is also recommended to back up the data to a physical copy once a week or every two weeks. This would allow access to the data should Vince ever lose internet access.



Thank you for your consideration of the proposal. I would be happy to answer any questions you may have at this time.

REFERENCES

AWS. (2019). *Amazon Aurora*. Amazon Web Services, Inc. https://aws.amazon.com/rds/aurora/

Berecki, B. (2019, June 28). *6 Data Protection Laws for US Organizations*. Endpoint Protector; CoSoSys Ltd. https://www.endpointprotector.com/blog/6-data-protection-laws-for-us-organizations/

Klosowski, T. (2021, September 6). *The State of Consumer Data Privacy Laws in the US (And Why It Matters)*. Wirecutter, Inc.; The New York Times. https://www.nytimes.com/wirecutter/blog/state-of-privacy-laws-in-us/

Microsoft. (2019). *SQL Server 2019*. Microsoft. https://www.microsoft.com/en-us/sql-server/sql-server-2019

MySQL. (2000). *MySQL*. MySQL; Oracle Corporation. https://www.mysql.com/

PostgreSQL. (2019). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. PostgreSQL; The PostgreSQL Global Development Group. https://www.postgresql.org/