

Store Database

Contents

[Part One 3](#_Toc114527273)

[Summary 3](#_Toc114527274)

[Stakeholders 3](#_Toc114527275)

[Business Rules 3](#_Toc114527276)

[Data Questions 4](#_Toc114527277)

[Conceptual Model 4](#_Toc114527278)

[Logical Model 5](#_Toc114527279)

[Part Two 5](#_Toc114527280)

[Data Definition Language: Creating Tables and Constraints 5](#_Toc114527281)

[Data Manipulation Language: Inserting Data 8](#_Toc114527282)

[Answering Data Questions 12](#_Toc114527283)

[Programming Objects 16](#_Toc114527284)

[User Interface 17](#_Toc114527285)

[Reflection 22](#_Toc114527286)

# Part One

# Summary

3 Boys Enterprises Laundromat is a family-owned Laundromat located in Spring Valley, New York, and has been in operation since 2010. Since then, the business has evolved drastically, but still resorts to inefficient methods for tracking customer purchases, inventory/stock, drop-off service customer information, employee data, as well as the most recent addition of customer laundromat cards. To evolve the business further and improve the tracking of relevant data, I will build an all-encompassing data base to contain it for storage and retrieval.

# Stakeholders

* 3 Boys Enterprises Management/Employees
  + People within the business who perform many operations for the store
* 3 Boys Enterprises Customers
  + Recurring and new consumers of the business
* Suppliers
  + Companies who provide inventory for the laundromat

# Business Rules

* Every customer who washes and dries is will be identified by the ID on their laundromat card, but first/last name, address, email, phone number are optional identifiers (If card is lost, it is convenient to have all info present).
* Customers who utilize drop off service are required to be identified by their first/last name, and phone number. Address, email address, and laundromat card ID are optional. Weight of clothes will also be calculated
* Employees are required to have their first/last name, address, phone number, and email address available in database.
* Employees manage stock by documenting sales (transactions) made to customers
* Suppliers on delivery day give employees the product to be stored or stocked on shelves

# Data Questions

* How much do we generate from washers and dryers on a given day?
  + Track a card’s payment on given day by its ID
* What is the average weight of drop-off service clothes?
  + Use customer drop-off weight information to calculate average for reasonable sample size
* What employees complete the most transactions?
  + View Transactions made to customers and which employee assisted
* More!

# Conceptual Model

Diagram, schematic

Description automatically generated

# Logical Model

(Please Zoom in for Higher Quality)

Diagram, schematic

Description automatically generated

# Part Two

# Data Definition Language: Creating Tables and Constraints

--Creating the Supplier Table

CREATE TABLE Supplier(

--Columns for Supplier Table

SupplierID int identity,

CompanyName varchar(50) not null,

DeliveryDate datetime not null default GetDate(),

SupplyPrice int not null,

SupplyOrderID int not null,

--Constraints on the Supplier Table

CONSTRAINT PK\_Supplier PRIMARY KEY (SupplierID)

)

--Creating the SupplyOrder Table

CREATE TABLE SupplyOrder(

--Columns for SupplyOrder Table

SupplyOrderID int identity,

OrderDescription varchar(200) not null,

SupplyPrice int not null,

ItemID int not null,

EmployeeID int not null,

--Constraints on the Supply Order Table

CONSTRAINT PK\_SupplyOrder PRIMARY KEY (SupplyOrderID)

)

--Adding Constraint for SupplyOrderID FK on Supplier Table

ALTER TABLE Supplier

ADD CONSTRAINT FK\_SupplyOrderID FOREIGN KEY (SupplyOrderID) REFERENCES SupplyOrder(SupplyOrderID)

--Creating the Employee Table

CREATE TABLE Employee(

--Columns for the Employee Table

EmployeeID int identity,

EmployeeFirstName varchar(30) not null,

EmployeeLastName varchar(30) not null,

EmployeePhoneNumber varchar(15) not null,

EmployeeEmailAddress varchar(30) not null,

EmployeeAddress varchar(30) not null,

JobTitle varchar(20) not null,

ManagerID int,

--Constraints on the Employee Table

CONSTRAINT PK\_Employee PRIMARY KEY (EmployeeID),

CONSTRAINT U1\_Employee UNIQUE (EmployeePhoneNumber),

CONSTRAINT U2\_Employee UNIQUE (EmployeeEmailAddress)

)

--Adding Constraint for EmployeeID FK on SupplyOrder Table

ALTER TABLE SupplyOrder

ADD CONSTRAINT FK\_EmployeeID FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

--Creating the Item Table

CREATE TABLE Item(

--Columns for the Item Table

ItemID int identity,

Brand varchar(20) not null,

SalesPrice int not null,

Quantity int not null,

--Constraints on the Item Table

CONSTRAINT PK\_Item PRIMARY KEY (ItemID)

)

--Adding Constraint for ItemID FK on SupplyOrder Table

ALTER TABLE SupplyOrder

ADD CONSTRAINT FK\_ItemID FOREIGN KEY (ItemID) REFERENCES Item(ItemID)

--Creating the SaleOrder Table

CREATE TABLE SalesOrder(

--Columns for the SalesOrder Table

SalesOrderID int identity,

ItemID int not null,

EmployeeID int not null,

CustomerID int,

--Constraints on the SalesOrder Table

CONSTRAINT PK\_SalesOrder PRIMARY KEY (SalesOrderID),

CONSTRAINT FK1\_SalesOrder FOREIGN KEY (ItemID) REFERENCES Item(ItemID),

CONSTRAINT FK2\_SalesOrder FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

)

--Creating the Customer Table

CREATE TABLE Customer(

--Columns for the Customer Table

CustomerID int,

CustomerFirstName varchar(30),

CustomerLastName varchar(30),

CustomerAddress varchar(40),

CustomerEmailAddress varchar(30),

LaundromatCardID int

--Constraints on the Customer Table

CONSTRAINT PK\_Customer PRIMARY KEY (CustomerID)

)

--Adding Constraint for CustomerID on the SalesOrder Table

ALTER TABLE SalesOrder

ADD CONSTRAINT FK\_CustomerID FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

--Creating the LaundromatCard Table

CREATE TABLE LaundromatCard(

--Columns for the LaundromatCard Table

LaundromatCardID int,

CardName varchar(50),

CardPin int not null,

--Constraints on the LaundromatCard Table

CONSTRAINT PK\_LaundromatCard PRIMARY KEY (LaundromatCardID),

CONSTRAINT U1\_LaundromatCard UNIQUE (CardPin)

)

--Creating the Washer Table

CREATE TABLE Washer(

--Columns for the Washer Table

WasherID int identity,

WasherSize varchar(30) not null,

WasherPrice varchar(30) not null,

Duration varchar(30) not null,

LaundromatCardID int not null,

--Constraints on the Washer Table

CONSTRAINT PK\_Washer PRIMARY KEY (WasherID),

CONSTRAINT FK1\_Washer FOREIGN KEY (LaundromatCardID) REFERENCES LaundromatCard(LaundromatCardID)

)

--Creating the Dryer Table

CREATE TABLE Dryer(

--Columns for the Dryer Table

DryerID int identity,

Size varchar(30) not null,

PricePerMinute varchar(20) not null,

LaundromatCardID int,

--Constraints on the Dryer Table

CONSTRAINT PK\_Dryer PRIMARY KEY (DryerID),

CONSTRAINT FK1\_Dryer FOREIGN KEY (LaundromatCardID) REFERENCES LaundromatCard(LaundromatCardID)

)

--Creating ther DropOffService Table

CREATE TABLE DropOffService(

--Columns for the DropOffService Table

ServiceID int identity,

FirstName varchar(30) not null,

LastName varchar(30) not null,

PhoneNumber varchar(15) not null,

Address varchar(40),

LaundromatCardID int,

ClothesWeight varchar(30) not null,

ClothesPrice int not null,

EmployeeID int not null,

--Constraints on the DropOffService Table

CONSTRAINT PK\_DropOffService PRIMARY KEY (ServiceID),

CONSTRAINT FK1\_DropOffService FOREIGN KEY (LaundromatCardID) REFERENCES LaundromatCard(LaundromatCardID),

CONSTRAINT FK2\_DropOffService FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

)

# Data Manipulation Language: Inserting Data

--Data Manipulation Language - Inserting Data

--Insert Data for Employee Table

insert into Employee (EmployeeFirstName, EmployeeLastName, EmployeePhoneNumber, EmployeeEmailAddress, EmployeeAddress, JobTitle, ManagerID ) values ('Vivyanne', 'Hanhart', '400-563-4256', 'vhanhart0@sfgate.com', '37 Del Mar Plaza', 'Manager', 1);

insert into Employee (EmployeeFirstName, EmployeeLastName, EmployeePhoneNumber, EmployeeEmailAddress, EmployeeAddress, JobTitle) values ('Adair', 'Websdale', '814-174-0394', 'awebsdale1@forbes.com', '8692 Mendota Road', 'Assistant Manager');

insert into Employee (EmployeeFirstName, EmployeeLastName, EmployeePhoneNumber, EmployeeEmailAddress, EmployeeAddress, JobTitle) values ('Celestine', 'D''Aulby', '412-854-6755', 'cdaulby2@cmu.edu', '4 Melvin Lane', 'Assistant Manager');

insert into Employee (EmployeeFirstName, EmployeeLastName, EmployeePhoneNumber, EmployeeEmailAddress, EmployeeAddress, JobTitle) values ('Nerty', 'Killeen', '225-876-7827', 'nkilleen3@google.ru', '68 Comanche Way', 'Sales Associate');

insert into Employee (EmployeeFirstName, EmployeeLastName, EmployeePhoneNumber, EmployeeEmailAddress, EmployeeAddress, JobTitle) values ('Cary', 'MacGilfoyle', '652-880-0465', 'cmacgilfoyle4@cnbc.com', '056 Darwin Plaza', 'Sales Associate');

SELECT \* FROM Employee

--Insert Data for SupplyOrder Table

insert into SupplyOrder (CompanyName, DeliveryDate, OrderDescription, SupplyPrice, EmployeeID) values ('Colgate-Palmolive', '8/4/2022', 'Medium sized order', 490.55, 1);

insert into SupplyOrder (CompanyName, DeliveryDate, OrderDescription, SupplyPrice, EmployeeID) values ('Protector & Gamble', '8/10/2022', 'Smaller than medium sized order', 341.43, 1);

insert into SupplyOrder (CompanyName, DeliveryDate, OrderDescription, SupplyPrice, EmployeeID) values ('Aura Detergent', '8/10/2022', 'Barrel of soap', 238.80, 1);

insert into SupplyOrder (CompanyName, DeliveryDate, OrderDescription, SupplyPrice, EmployeeID) values ('The Clorox Company', '8/16/2022', 'Please do not get this on colored clothes', 192.71, 2);

insert into SupplyOrder (CompanyName, DeliveryDate, OrderDescription, SupplyPrice, EmployeeID) values ('Protector & Gamble', '8/25/2022', 'Hey, us again!', 300.14, 2);

SELECT \* FROM SupplyOrder

--Insert Data for Item Table

insert into Item (Brand, SalesPrice, Quantity) values ('Medium Red Tide', 11.00, 80);

insert into Item (Brand, SalesPrice, Quantity) values ('Medium Gain', 10.00 , 95);

insert into Item (Brand, SalesPrice, Quantity) values ('Small Suavitel Blue', 2.25, 9);

insert into Item (Brand, SalesPrice, Quantity) values ('Small Suavitel White', 2.25, 41);

insert into Item (Brand, SalesPrice, Quantity) values ('Small Clorox', 2.25, 81);

insert into Item (Brand, SalesPrice, Quantity) values ('Medium Clorox', 3.50, 58);

insert into Item (Brand, SalesPrice, Quantity) values ('Small Roma', 1.50, 11);

insert into Item (Brand, SalesPrice, Quantity) values ('Medium Roma', 3.00, 15);

insert into Item (Brand, SalesPrice, Quantity) values ('Large Aura', 10.00, 6);

insert into Item (Brand, SalesPrice, Quantity) values ('Shout Spray', 10.00, 1);

SELECT \* FROM Item

--Insert Data for LaundromatCard Table

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (916679463, 'cmagill0', 1204);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (568424468, 'wmcginnell1', 4290);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (356331589, 'jrudman2', 8524);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (881872442, 'lwaddams3', 8031);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (160890131, 'jbonefant4', 1446);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (285754330, 'myakunchikov5', 8820);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (736784726, 'lcaress6', 9058);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (450588115, 'rgodain7', 9094);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (174372923, 'ekirkwood8', 3335);

insert into LaundromatCard (LaundromatCardID, CardName, CardPin) values (141613498, 'kpoll9', 4571);

SELECT \* FROM LaundromatCard

--Insert Data for Customer Table

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress, LaundromatCardID) values (1, 'Debra', 'Place', '37 Erie Plaza', 'dplace0@usatoday.com', 916679463);

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress, LaundromatCardID) values (2, 'Lorianne', 'Frazer', '4485 Westridge Place', 'lfrazer1@flickr.com', 568424468);

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress, LaundromatCardID) values (3, 'Estel', 'Harrow', '5 Schiller Road', 'eharrow2@eepurl.com', 356331589);

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress, LaundromatCardID) values (4, 'Markus', 'Reddan', '6924 Marcy Street', 'mreddan3@baidu.com', 881872442);

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress, LaundromatCardID) values (5, 'Ronny', 'Cleal', '3 Schurz Center', 'rcleal4@latimes.com', 160890131);

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress) values (6, 'Ole', 'Byrde', '5 Loomis Crossing', 'obyrde5@imageshack.us');

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress) values (7, 'Jamil', 'Milton-White', '3012 Declaration Plaza', 'jmiltonwhite6@about.com');

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress) values (8, 'Bernelle', 'Priestner', '9 Thackeray Alley', 'bpriestner7@cam.ac.uk');

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress) values (9, 'Dierdre', 'Older', '55251 Hoffman Drive', 'dolder8@twitpic.com');

insert into Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerAddress, CustomerEmailAddress) values (10, 'Henderson', 'Chiswell', '87 Lighthouse Bay Drive', 'hchiswell9@salon.com');

SELECT \* FROM Customer

--Insert Data for SalesOrder Table

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (1, 4, 1);

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (1, 4, 3);

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (3, 4, 4);

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (3, 4, 5);

insert into SalesOrder (ItemID, EmployeeID) values (5, 5);

insert into SalesOrder (ItemID, EmployeeID) values (9, 5);

insert into SalesOrder (ItemID, EmployeeID) values (8, 5);

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (10, 5, 7);

insert into SalesOrder (ItemID, EmployeeID) values (2, 1);

insert into SalesOrder (ItemID, EmployeeID, CustomerID) values (12, 2, 9);

SELECT \* FROM SalesOrder

--Insert Data for Washer Table

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Small', 3.00, 25, 141613498);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Small', 3.00, 25, 160890131);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Small', 3.00, 25, 160890131);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Medium', 5.50, 28, 141613498);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Medium', 5.50, 28, 285754330);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Medium', 5.50, 28, 285754330);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Large', 6.50, 30, 450588115);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Large', 6.50, 30, 736784726);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Extra Large', 10.00, 35, 881872442);

insert into Washer (WasherSize, WasherPrice, Duration, LaundromatCardID) values ('Extra Large', 10.00, 35, 916679463);

SELECT \* FROM Washer

--Insert Data for Dryer Table

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 916679463);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 916679463);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 450588115);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 450588115);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 736784726);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Large', 0.25, 141613498);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Extra Large', 0.50, 141613498 );

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Extra Large', 0.50, 141613498);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Extra Large', 0.50, 285754330);

insert into Dryer (Size, PricePer10Minutes, LaundromatCardID) values ('Extra Large', 0.50, 285754330);

SELECT \* FROM Dryer

--Insert Data for DropOffService Table

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Marlee', 'Trenbey', '127-747-1690', '3323 Fieldstone Pass', 50, 50, 4);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Forrest', 'Shardlow', '218-618-5833', '02824 Northport Court', 80, 80, 2);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Blondelle', 'Mugg', '602-382-7755', '1 Northview Way', 93, 93, 3);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Stanly', 'Crofthwaite', '860-502-7896', '8006 Heffernan Center', 62, 62, 1);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Celka', 'Nassie', '803-959-7626', '443 Tennyson Lane', 57, 57, 1);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, ClothesWeight, ClothesPrice, EmployeeID) values ('Ossie', 'Cordoba', '309-935-9536', '70 Hudson Court', 100, 100, 3);

insert into DropOffService (FirstName, LastName, PhoneNumber, Address, LaundromatCardID, ClothesWeight, ClothesPrice, EmployeeID) values ('Evelyn', 'Nuttey', '445-341-5334', '9 Express Park',141613498, 28, 28, 2);

SELECT \* FROM DropOffService

# 

# Answering Data Questions

**How revenue did we generate from washers and dryers today?**

We can answer this by tracking a card’s payment on given day by its ID

--Convert Washer and Dryer to number instead of string to do math

ALTER TABLE Washer

ALTER COLUMN WasherPrice decimal;

ALTER TABLE Dryer

ALTER COLUMN PricePer10Minutes decimal;

--Displaying washers used in given day, and which card made these purchases

SELECT

Washer.LaundromatCardID,

Washer.WasherSize,

Washer.WasherPrice

FROM Washer

Table

Description automatically generated

SELECT SUM(WasherPrice) AS WasherTotalForDay

FROM Washer



--Displaying Dryers used in given day, and which card made these purchases

SELECT

Dryer.LaundromatCardID,

Dryer.Size,

Dryer.PricePer10Minutes

FROM Dryer

Table

Description automatically generated

Explanation for this field: PricePer10Minutes counts a transaction as adding 10 minutes to the dryer. So LaundromatCardID “141613498” likely used 20 minutes worth of time on an Extra-Large Dryer. The original values were 0.25 for Large, and 0.50 for Extra Large, but the conversion made the decimals round the numbers off. I am not as concerned with the outcome as much as I am with the query working properly. Next time, I will have to ensure my logical model is constructed better to prevent issues like this.

A conclusion we can draw from this data question (despite numbers being messed up for Dryer) is that we make more money from washers than dryers. We’d have to test multiple days to confirm whether this is true. With that, we can develop an average for days, and further subset that into specific days of the week.

**What is the average weight of drop-off clothes (In Pounds/Lbs)?**

--Displaying all the Drop-Offs we have on record and which employee did the service

SELECT

DropOffService.FirstName,

DropOffService.LastName,

DropOffService.ClothesWeight,

DropOffService.ClothesPrice,

Employee.EmployeeID

FROM DropOffService

JOIN Employee ON DropOffService.EmployeeID = Employee.EmployeeID

Table

Description automatically generated with medium confidence

--Convert Clothes Weight to number instead of string to do math

ALTER TABLE DropOffService

ALTER COLUMN ClothesWeight INT;

SELECT AVG(ClothesWeight) AS AverageOfClothesWeight

FROM DropOffService



The average amount of weight for clothes in a drop off service is 67 pounds. This reveals some good information to us, as we know that drop offs are $1 per pound. So essentially according to our sample size, after about 7 drop offs completed, we can expect an average of $67.

**Which employee completes the most transactions?**

--Displaying all transactions on record with Employee Information

SELECT

SalesOrder.EmployeeID,

Employee.EmployeeFirstName,

Employee.EmployeeLastName,

SalesOrder.ItemID,

SalesOrder.CustomerID

FROM SalesOrder

JOIN Employee ON SalesOrder.EmployeeID = Employee.EmployeeID

Table

Description automatically generated

Just from visually looking at the output, we can count that EmployeeID 4 (Nerty Killeen), and EmployeeID 5 (Cary MacGiloyle) are tied for the most transactions completed on record. But if we had a lot more records, manually counting would not be practical. In that case, I’d solve it like this:

--Who sells the most?

SELECT EmployeeID, COUNT(\*) AS TotalEmployeeTransactions

FROM SalesOrder

GROUP BY EmployeeID

ORDER BY TotalEmployeeTransactions DESC

A picture containing table

Description automatically generated

If prices were included via a join from the item table, that would add another layer in which we could use to not only tell which employee had the most transactions, but who sold the most in terms of revenue generated. In hindsight, the price should’ve been included in the SalesOrder table in some fashion.

**What does our inventory of items look like?**

--Inventory of Items

SELECT

Item.Brand,

Item.Quantity

FROM Item

A picture containing table

Description automatically generated

**If we had more records, it would be more difficult to get a grasp of what needs a restock. How do we determine what we should prioritize in terms of replenishing stock?**

SELECT

Item.Brand,

Item.Quantity

FROM Item

ORDER BY Quantity

**Graphical user interface, table

Description automatically generated**

Now that things are ordered, we can prioritize our needs better.

**What happens if we want to check a specific brand or item’s quantity? In a large table this can be difficult to manually see.**

Let’s say I wanted to see the quantity of my Suavitel products only:

--Find specific brands

SELECT

Item.Brand,

Item.Quantity

FROM

Item

WHERE

Item.Brand LIKE '%Suavitel%'

Graphical user interface

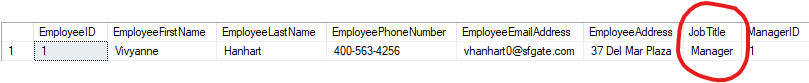
Description automatically generated

These are useful methods in filtering our inventory and helping determine what needs to be restoked with priority in mind.

# Programming Objects

Utilizing a stored procedure to update employee’s Job Title

Before:



After:

--Update an employees Job Title

CREATE PROCEDURE UpdateTitle(@EmployeeFirstName varchar(30), @NewJobTitle varchar(20))

AS

BEGIN

UPDATE Employee SET JobTitle = @NewJobTitle

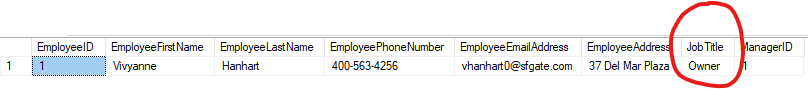
WHERE EmployeeFirstName = @EmployeeFirstName

END

GO

EXEC UpdateTitle 'Vivyanne', 'Owner'

SELECT \* FROM Employee WHERE EmployeeFirstName = 'Vivyanne'



This is a good example of how a programming object can be utilized in my database. Functions can be used to find specific values, and queries can be used like above in a more efficient manner than manually them. To create intricate ones however, I’d have to first reoptimize some elements of my database.

# User Interface

I chose to use Access for my User Interface, and reestablished the relationships between the Fields:

Diagram

Description automatically generated

In Access, I’ll allow the front-end user to document a new entry for a drop off service by creating a new form and letting them manually enter the data. Afterwards, it should update the database in real time.

DropOffService Records before:

SELECT \* FROM DropOffService

Graphical user interface, text, application, table

Description automatically generated

Our Front-End User is Nerty Killeen and she just received a new drop off to complete. Here is what it looks like on her side:

Graphical user interface, table

Description automatically generated

After she entered some new data:

Table

Description automatically generated

--Checking to see if Database updated from Access

SELECT \* FROM DropOffService

Graphical user interface

Description automatically generated with medium confidence

As we can see, a new record has been added and we know which employee completed the task based on their ID.

Next, we’d like to observe an employee complete a SupplyOrder and enter a new record.

SupplyOrder Before:

SELECT \* FROM SupplyOrder

Text

Description automatically generated with medium confidence

This is what our employee sees on their end: Graphical user interface, application

Description automatically generated

Nerty enters some more data:

Graphical user interface, application, table

Description automatically generated

--Checking to see if Database updated from Access

SELECT \* FROM DropOffService

Application

Description automatically generated with medium confidence

The record has been added and we once again know who it based on their employee ID. If we wanted to see the name associated, we could enter a SELECT Query with a join.

Finally, we will see Front-end users utilize the report feature to turn a form into a report.

This is the form we’ve decided to go with, and it represents SalesOrder combined with price, the employee who sold it, and other relevant information.

Graphical user interface

Description automatically generated with medium confidence

When converted into a report, this is the report view:

Graphical user interface

Description automatically generated

This can make the results of a form more printer friendly, and more readable overall. There is also an ability to personally customize your own report style, but I went with the default option.

# Reflection

**What assumptions did you have at the start of your project that changed by the end? Think in terms of both your own problem domain as well as your knowledge of the process.**

The assumptions I had at the start of the project was that it would be less time consuming that it ended up being. This likely has a direct correlation with how many tables I chose to incorporate, but I think the excess time I spent repeating the same processes helped me better understand elements of SQL more in depth. I also presumed it would be very difficult, and while that is certainly subjective, I think it had just the right amount of challenge for me. By no means did I fly through it, but I also had to think about my decisions a lot. With that being said, there were still improvements to be had, but I know I will be better from the experience.

**The next time you do this, what will be different?**

Preparation will be different. Mainly in terms of creating better logical and conceptual models, I underestimated their importance and thought I’d just upgrade things along the way if need be. There were some major changes I’ve made to the final submission, but there were other things that could not be changed due to how much of what I’d completed relied on that foundation. It worked but could have been better in my opinion.

**Regardless of whether you go through these steps again, how do you think it will inform your approach to data as an information professional?**

It’ll be important to my approach I’d think. These steps place an underappreciated aspect of the conceptual steps to understanding why things need to be the way they are. Data needs to be good and make sense. Moving forward I’ll certainly be more cautious of how I work with data, especially if I am creating it from scratch.

# Summary

With this being my first time creating a database from scratch by myself, I think I did a good job. Not because it was created perfectly, but because of the areas in which I saw I could’ve improved as I went along had I understood SQL better. While everything is serviceable and works, there were things I could’ve done that would’ve made certain elements more efficient. Some of the potential improvements I learned at the “point of no return” include combining certain elements on the same table instead of having to later join them for the results to make sense, being more careful identifying my datatypes to prevent later ALTER statements, being more mindful of what I’m naming columns so that it not only makes sense to me but others, and overall, just taking more time in the planning phases instead of adapting when I run into problems. There were even some things I wanted to try but refrained because I not want to risk breaking something that already works.

If there wasn’t a deadline for my project, what I’ve completed would not be the final draft. This is not just because of the obvious lack of records for the sake of the project, but because I’ve thought of so many improvements, and different ways of handling things if I were to start from the beginning. The project was enjoyable for me as I was trying to solve real problems relevant to my life while not knowing much about the process (from experience). It was an eye-opening experience, and I think even accurate, to what real database designers are doing in the work force. I hope to continue honing my skills/knowledge around SQL and the Database process so I too can solve real problems in the workplace.