IST 659 M403

Course Project | 5/10/2019

A Production History Database

Gregory Richardson

# Part 1 – Database Design

# Summary

This project seeks to create a database to track daily plant performance. Right now, I am tracking this data (painstakingly) in a series of flat Excel files. Though they have done what I need them to do, updating them requires a lot of redundant data entry that is extremely time-consuming. In addition, it is very hard to analyze data in ways that the pages weren’t set up to track initially. My goal through this project is to develop a relational database that will help streamline data entry while allowing me to parse and analyze the data in various ways.

# Data Narrative

I am tracking the performance data for a textile production facility. We produce products to order. That is, we do not produce a product until a customer orders it specifically. There are hundreds of different products, and while most products are unique to a single customer, there are some that are more generic and may be made for more than 1 customer. The plant operates on a 24-7 schedule, with two 12-hour shifts a day. There are 4 shifts (A, B, C, and D) that work on a rotating schedule.

Products can be loosely grouped into two categories – Fine Denier and Heavy Denier. This distinction mostly determines how a product is manufactured. Fine Denier products are made using a two-stage system – running first through Spinning and then through Finishing. Heavy Denier products, on the other hand, are only processed through a Finishing step. There are 3 Spinning machines and 4 Finishing machines where a product can be run, and (for the purposes of this database at least) products can be run on any of the equipment.

Multiple products can run on a production line during a single shift. For each product run on a production line, there are certain metrics that are logged. These metrics are slightly different for Spinning and Finishing.

For Spinning, the amount of total pounds run through the machine is tracked, along with the number of pounds of any Mix or Waste produced. For Finishing, we sort of track the reverse. We record the total pounds of good (Q1) material produced. We also track the amount of Mix, Waste, and Off-Grade produced as well. Occasionally, Finishing lines will be used to process materials made in R&D, as well as other non-production tasks like Feedback (repackaging material). These weights need to be recorded as well, but their metrics are excluded from a shift’s overall performance.

# Data Questions

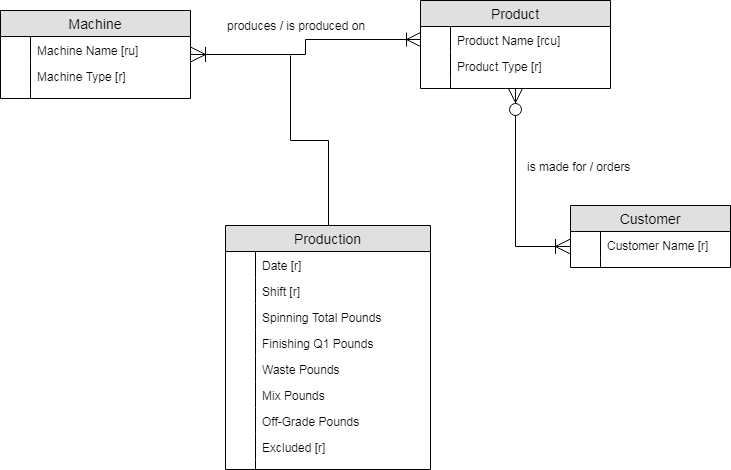
There are literally dozens of questions I use this data to answer on a routine basis. Some examples of these questions follow:

* What is the plant performance for a given shift/day/week/month/year? This productivity number is measured by the following formula:

* What is the Yield for a specific Machine/Machine Type for a given time period or shift
* What are our best/worst performing products?
* Who are our most valuable customers (in terms of production volume..)?
* How many products are run during a given time period or shift?
* What is the Yield for a specific Product Type for a given time period or shift?

# Conceptual Model

The ERD model for this production database follows. A list and description of each of the tables/attributes are included below that.



|  |  |  |
| --- | --- | --- |
| **Entity** | **Attribute** | **Description** |
| Machine | Machine Name | Required. Unique. Specific equipment name that is unique to the equipment (“Line 1”, “Tower 2”, etc) |
|  | Machine Type | Required. There are 2 different types of equipment; Either Spinning or Finishing. |
| Customer | Customer Name | Required |
| Product | Product Name | Required. Unique. Composite of 3 Parts – Denier, Product Code, and Cut Length. |
|  | Product Type | Required. Determines which process flow is required; Either Fine Denier or Heavy Denier. |
| Production | Date | Required |
|  | Shift | Required. Either A, B, C, or D shift |
|  | Spinning Total Pounds | Only used for Spinning product/process. Number of total pounds input into system |
|  | Finishing Q1 Pounds | Only used for Finishing product/process. Number of total good pounds output from system |
|  | Waste Pounds | Pounds of waste produced for a product/process. Used in both Spinning and Finishing |
|  | Mix Pounds | Pounds of mix produced for a product/process. Used in both Spinning and Finishing |
|  | Off-Grade Pounds | Pounds of Off-Grade produced for a product/process. Only used in Finishing processes. |
|  | Excluded | Required. Identifies any R&D or rework products for exclusion from production reporting. |

# Normalized Model

The first major change in Normalized Model from the Conceptual model is a new associative table has been added called shift\_production. This table houses the relationship between products and machines, and stores the key production measurements. In addition, an associative table has been created between product and customer, and this linkage is associated with the shift production.

The composite attribute Product Name was broken down into its 3 components – Denier, Product Code, and Cut Length. These were all made varchar fields, even though they are numeric since there’s no practical reason to do math on any of them.

For the measurement fields in the Shift Production table, I chose to use the integer data types. This is as much precision as we require – our current process rounds weight to the nearest whole pound already.

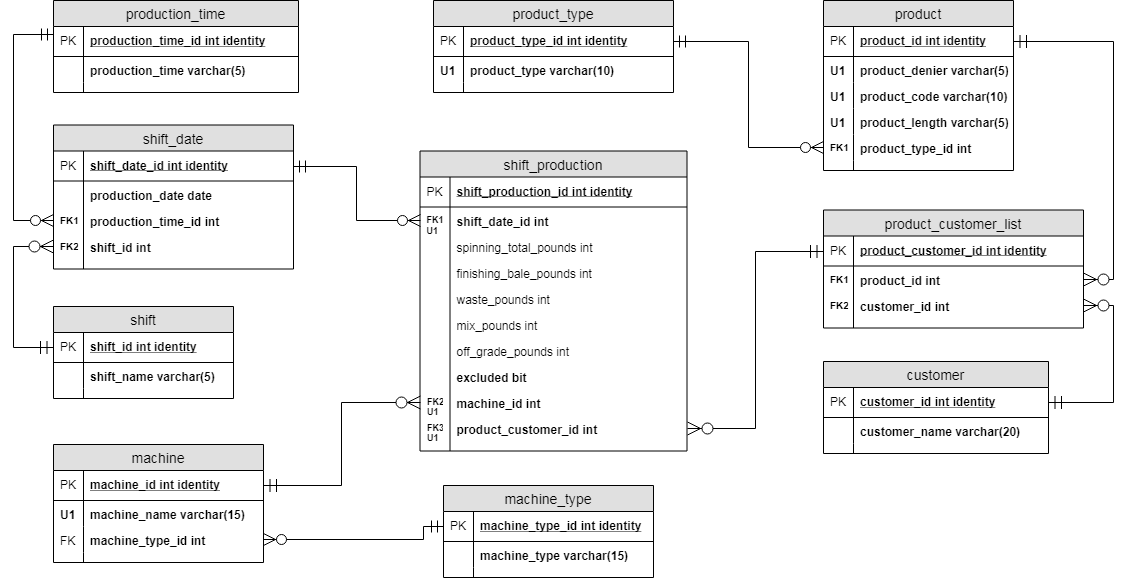
As described above, the Excluded field indicates any product that should be removed from any daily/weekly (etc) production report. This is often the case for R&D products, but may also be for rework that would otherwise skew the metrics. This field only needs to be a simple Boolean true or false, so I used the bit data type.

Each entity will use a surrogate key serving as Primary Key, and which will link the various tables together.

I made several decisions while normalizing my model in order to protect date integrity. I created new tables to house the entries for Shift, as well as for the Production Time (Day or Night), Product and Machine Type. These choices were made to minimize the chance for mistypes when entering data (like somebody typing “Nite” instead of “Night”) that would complicate data analysis.

Another new entity, Shift Date was created while normalizing. Because the Shift attribute is dependent on the combination of Production Date and Production Time, a new table was created to house that relationship.

There are a number of Uniqueness constraints as well. For instance, the combination of all three fields making up the Product Name is Unique – there should not be multiple versions of the name. In addition, in the Shift Production table, I specified a uniqueness constraint over multiple fields – the Shift Date, the Machine, and the Product/Customer. Because the data in this field may be entered by different people at different times, I wanted to ensure no accidental duplication of records with each only containing partial information.



# Part 2 – Database Implementation

# Physical Database Design

The code for the database table creation is listed below. Tables were created and linked as listed above in the normalized model.

-- Part 1: Table Creation

-- Creating Production Time table

-- This table is a lookup table housing the 2 unique daily shifts - Day and Night

CREATE TABLE production\_time (

production\_time\_id int identity PRIMARY KEY,

production\_time varchar(5) not null UNIQUE

)

-- Creating Shift Name table

-- This table is lookup table housing the unique shift names (A, B, C, and D)

CREATE TABLE shift\_name (

shift\_name\_id int identity PRIMARY KEY,

shift\_name varchar(5) not null UNIQUE

)

-- Creating Machine Type table

-- This table is lookup table housing the unique Machine Types (Spinning, Finishing)

CREATE TABLE machine\_type (

machine\_type\_id int identity PRIMARY KEY,

machine\_type varchar(15) not null UNIQUE

)

-- Creating Product Type table

-- This table is lookup table housing the unique Product Types (Fine Denier, Heavy Denier)

CREATE TABLE product\_type (

product\_type\_id int identity PRIMARY KEY,

product\_type varchar(15) not null UNIQUE

)

-- Creating Customer table

-- This table houses the names (and potentially other information in the future) of our customers who order product

CREATE TABLE customer (

customer\_id int identity PRIMARY KEY,

customer\_name varchar(20) not null

)

/\*

Creating Machine table

- This table houses the information (including name and type) of all of our production machines

- Each machine entry also includes an id referencing whether it is Spinning or Finishing

- There is a Unique constraint for entries based on Name AND type. ie - There can be 2

Machine #2's, but only one Machine #2 that is for Spinning

\*/

CREATE TABLE machine (

machine\_id int identity PRIMARY KEY,

machine\_name varchar(15) not null,

machine\_type\_id int not null FOREIGN KEY REFERENCES machine\_type(machine\_type\_id),

CONSTRAINT u1\_machine UNIQUE (machine\_name, machine\_type\_id)

)

/\*

Creating Product table

- This table houses information regarding our products - including the product code and product type

- Each entry also includes an id referencing its Product Type (Fine or Heavy Denier)

- Each item name is a combination of Denier, Product Code, and Product Length, and that combination

of 3 items is Unique.

\*/

CREATE TABLE product (

product\_id int identity PRIMARY KEY,

product\_denier varchar(5) not null,

product\_code varchar(10) not null,

product\_length varchar(5) not null,

product\_type\_id int not null FOREIGN KEY REFERENCES product\_type(product\_type\_id),

CONSTRAINT U1\_product UNIQUE (product\_denier, product\_code, product\_length)

)

/\*

Creating Shift Date table

- This table houses records of which shift was working a specific day and time

- Each entry includes an id referencing whether the shift is Day or Night

- Each entry includes an id referencing the Shift working (A, B, C or D)

- There is a constraint here that says only 1 Shift can work a specific Day/Shift combo.

That is - A and B shifts cannot BOTH work 1/1/2019 Day Shift...

\*/

CREATE TABLE shift\_date (

shift\_date\_id int identity PRIMARY KEY,

production\_date date not null,

production\_time\_id int not null FOREIGN KEY REFERENCES production\_time(production\_time\_id),

shift\_name\_id int not null FOREIGN KEY REFERENCES shift\_name(shift\_name\_id),

CONSTRAINT U1\_shift\_date UNIQUE (production\_date, production\_time\_id)

)

/\*

Creating Product Customer table

- This table lists the products ordered by a specific customer

- Each customer can have multiple products and each product can have

multiple customers, but a specific customer/product combination can only

appear in this list once.

\*/

CREATE TABLE product\_customer\_list (

product\_customer\_id int identity PRIMARY KEY,

product\_id int not null FOREIGN KEY REFERENCES product(product\_id),

customer\_id int not null FOREIGN KEY REFERENCES customer(customer\_id),

CONSTRAINT U1\_product\_customer UNIQUE (product\_id, customer\_id)

)

/\*

Creating Shift Production

- This table houses the yield measurements for a given production shift, machine, and product

- Each entry must include: an id for the Shift/Date that created it, the id for the machine producing it,

and an id for the product produced and customer it was produced for.

- There are optional fields for inputting the pounds created, as well as various off-grade pounds.

- There is an additional required field for "Excluded". Some products are R&D products that are not included in

Production calculations. This will help filter them out when doing those views.

- There is a constraint on this table that a product can only appear on a machine once in a specific date/shift. This is

to avoid potential duplication/accidental separation of data.

\*/

CREATE TABLE shift\_production (

shift\_production\_id int identity PRIMARY KEY,

shift\_date\_id int not null FOREIGN KEY REFERENCES shift\_date(shift\_date\_id),

spinning\_total\_pounds int,

finished\_Q1\_pounds int,

waste\_pounds int DEFAULT 0,

mix\_pounds int,

off\_grade\_pounds int,

excluded bit not null,

machine\_id int not null FOREIGN KEY REFERENCES machine(machine\_id),

product\_customer\_id int not null FOREIGN KEY REFERENCES product\_customer\_list(product\_customer\_id),

CONSTRAINT U1\_shift\_production UNIQUE (shift\_date\_id, machine\_id, product\_customer\_id)

)

# Inserting Initial Data

The next step was to insert some initial data into the database so that I could begin working on creating appropriate views, functions, and procedures. I started with populating the lookup tables with their respective entries. These items are not likely to change in the near future and were simply coded in manually.

When it came to customer, product, and daily production data however, there was simply too much to manually code in. Instead, I took an Excel file (essentially the current system this database will hopefully replace), cleaned the data and imported the worksheets in as temporary tables. I then copied the needed data from those temporary tables into the permanent tables, looking up required primary key ids along the way, and then removed the temporary tables from the database.

A copy of the temporary data is not included here, but can be provided upon request.

/\* Part 2

Part 2A : Adding Basic Information to Lookup Tables

Tables: Machine Type, Product Type, Shift Name, Machine

\*/

-- The Machine Type table is a lookup table including only 2 values - Spinning and Finishing.

INSERT INTO machine\_type

(machine\_type)

VALUES

('Spinning'),

('Finishing')

-- The Product Type table is a lookup table including only 2 values - Fine Denier and Heavy Denier

INSERT INTO product\_type

(product\_type)

VALUES

('Fine Denier'),

('Heavy Denier')

-- The Shift Name table includes the names of the 4 working shifts - A, B, C, and D

INSERT INTO shift\_name

(shift\_name)

VALUES

('A'),

('B'),

('C'),

('D')

-- The Production Time table is a lookup table including only 2 values - 'Day' and 'Night'

INSERT INTO production\_time

(production\_time)

VALUES

('Day'),

('Night')

-- There are 3 Spinning machines (labeled 1, 2, and 3) and 4 Finishing machines (labeled 1,2,3, and 4).

INSERT INTO machine

(machine\_name, machine\_type\_id)

VALUES

(1, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Spinning')),

(2, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Spinning')),

(3, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Spinning')),

(1, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Finishing')),

(2, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Finishing')),

(3, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Finishing')),

(4, (SELECT machine\_type\_id FROM machine\_type WHERE machine\_type LIKE 'Finishing'))

/\*

Part 2B : Adding Data From Imported Files

Tables: Product, Customer, Product\_Customer\_List, Shift\_Date, and Shift\_Production

Initial data was cleaned and made ready for the database in Excel. Data was imported into temporary tables.

This section is for inserting the data from the temporary tables to the live tables and then removing the temp tables

\*/

-- Inputting Product information from a cleaned Excel spreadsheet. The spreadsheet lists the Product Type (Fine or Heavy), so I used

-- subquery to input the appropriate type id into the Product table.

INSERT INTO product (product\_denier, product\_code, product\_length, product\_type\_id)

SELECT Denier, Product1, Cut\_Length, (SELECT product\_type\_id FROM product\_type WHERE product\_type LIKE Product$.Type) FROM Product$

-- Inserting a list of Customer names from a cleaned Excel spreadsheet imported into a temporary table.

INSERT INTO customer (customer\_name)

SELECT Customer FROM Customer$

-- Inserting a list of Products and their Customers from a cleaned Excel Spreadsheet imported into a temporary table.

-- 2 subqueries were used - one to get the product ID from the Product table, using the 3-part Product Code in the temporary table

-- The second subquery returns the Customer ID matching the appropriate customer in the Customer table.

INSERT INTO product\_customer\_list (product\_id, customer\_id)

SELECT

(SELECT product\_id FROM product WHERE product\_denier LIKE productCustomer$.Denier AND product\_code LIKE ProductCustomer$.Product1 AND product\_length LIKE ProductCustomer$.Code),

(SELECT customer\_id FROM customer WHERE customer\_name LIKE ProductCustomer$.Customer)

FROM ProductCustomer$

-- Inserting a list of Shifts worked on selected days. There are 2 shifts per day, each worked by 1 of 4 shifts.

-- 2 subqueries used to insert the data into to the Shift\_Date table. The first gets the Shift Name id from the

-- Shift\_Name table matching the appropriate Shift Name

-- The second subquery returns the Production Time Id from its lookup table that matches the appropriate Production Time.

INSERT INTO shift\_date (production\_date, shift\_name\_id, production\_time\_id)

SELECT date,

(SELECT shift\_name\_id FROM shift\_name WHERE shift\_name LIKE ShiftDate$.Shift),

(SELECT production\_time\_id FROM production\_time WHERE production\_time LIKE ShiftDate$.Time)

FROM ShiftDate$

/\*

Inserting SPINNING data into the Shift Production table. This table is where the magic happens.

There are 2 files which were brought in to temporary tables. One for Spinning and one for Finishing. The first INSERT

will bring in the data from Spinning. Several subqueries will be required. One to get the appropriate Shift Date ID

from the Date/Shift data in the table. Another subquery will get the appropriate Product/Customer id, and another to return

the appropriate Machine ID.

For Spinning, the fields required will be Spinning Total Pounds, Waste, and Mix. All items are included, so the exclusion bit will be a 0.

All other entries will be Null.

\*/

INSERT INTO shift\_production (shift\_date\_id, machine\_id, product\_customer\_id, spinning\_total\_pounds, mix\_pounds, waste\_pounds, excluded)

SELECT

(SELECT shift\_date\_id FROM shift\_date JOIN production\_time ON production\_time.production\_time\_id = shift\_date.production\_time\_id

WHERE production\_date = Spinning$.Date AND Spinning$.Time LIKE production\_time.production\_time),

(SELECT machine\_id FROM machine WHERE machine\_name LIKE Spinning$.Tower AND machine\_type\_id = 1),

(SELECT product\_customer\_id FROM product\_customer\_list

JOIN product ON product.product\_id = product\_customer\_list.product\_id

JOIN customer ON customer.customer\_id = product\_customer\_list.customer\_id

WHERE product\_denier LIKE Spinning$.Denier AND product\_code LIKE Spinning$.Product1

AND product\_length LIKE Spinning$.CutLength AND customer\_name LIKE Spinning$.Customer),

TotalPounds,

Mix,

Waste,

0

FROM Spinning$

/\* The following statement will add data from the Finishing temporary table. This is much the same as the Spinning table,

except here, the Spinning Pounds and Mix fields will be NULL.

\*/

INSERT INTO shift\_production (shift\_date\_id, machine\_id, product\_customer\_id, finished\_Q1\_pounds, off\_grade\_pounds, waste\_pounds, excluded)

SELECT

(SELECT shift\_date\_id FROM shift\_date JOIN production\_time ON production\_time.production\_time\_id = shift\_date.production\_time\_id

WHERE production\_date = Finishing$.Date AND Finishing$.Time LIKE production\_time.production\_time),

(SELECT machine\_id FROM machine WHERE machine\_name LIKE Finishing$.Location AND machine\_type\_id = 2),

(SELECT product\_customer\_id FROM product\_customer\_list

JOIN product ON product.product\_id = product\_customer\_list.product\_id

JOIN customer ON customer.customer\_id = product\_customer\_list.customer\_id

WHERE product\_denier LIKE Finishing$.Denier AND product\_code LIKE Finishing$.Code

AND product\_length LIKE Finishing$.CutLength AND customer\_name LIKE Finishing$.Customer),

FinishingPounds,

ISNULL(SubPounds, 0),

ISNULL(Waste, 0),

0

FROM Finishing$

-- Dropping temporary tables from the Database

DROP TABLE Customer$

DROP TABLE Finishing$

DROP TABLE Product$

DROP TABLE ProductCustomer$

DROP TABLE ShiftDate$

DROP TABLE Spinning$

# Functions and Stored Procedures

Several functions and stored procedures were created to ease typical functions the database will handle. For instance, there are procedures for creating new customers, new products, and new product-customer links. There is also a procedure for adding new date-shift combinations, and a larger function for creating new shift production data. Several functions were created to help quickly look up data such as product and machine ids. There are also functions that will compute typical data calculations – the Q1% - percent of good product made divided by total material produced - for a given time span. There is a smaller function that does essentially the same thing, but can be used to find the % Waste, % Mix, and % Off-Grade as well. Finally, a procedure was created to easily update the Waste for a specific shift-production. This is needed because Waste is often recorded later than the initial Production data so this procedure will allow the Waste to be updated as needed.

/\*

PART 3

Creating Functions for quickly recalling certain data and running common calculations. These functions include recalling

a Product ID given its component parts. There is also a function for summarizing data for Q1 calculations - the most basic use

of this database. An additional function takes the summarized table data and returns a Q1% metric for a given time period.

Creating Stored Procedures for inserting new data into relevant tables. These tables include:

New Shift/Date pairs, New Products, New Customers, New Product/Customer Pairs, and New Shift Production

Another Stored Procedure is created for Updating Waste on a specific Shift Production, as this tends to be done

at a later point than the intitial data entry.

\*/

-- Creating a Function for recalling a Product ID. The Function takes in the Denier, Code, and Cut Length and returns the

-- appropriate Product ID.

GO

CREATE FUNCTION ProdID (@denier varchar(5), @code varchar(10), @length varchar(5))

RETURNS int AS

BEGIN

DECLARE @prod AS int = 0

SET @prod = (SELECT product\_id FROM product WHERE product\_denier LIKE @denier AND product\_code LIKE @code AND product\_length LIKE @length)

RETURN @prod

END

-- Creating a Function to recall Date/Shift ID. Function takes in the Production Date and the Shift and returns the Shift\_Date ID.

GO

CREATE FUNCTION DateID (@date date, @shift varchar(5))

RETURNS int AS

BEGIN

DECLARE @dateID AS int=0

SET @dateID = (

SELECT shift\_date\_id FROM shift\_date

JOIN shift\_name ON shift\_date.shift\_name\_id = shift\_name.shift\_name\_id

WHERE production\_date = @date

AND @shift LIKE shift\_name

)

RETURN @dateID

END

-- Creating a Function to recall Machine ID. Function takes in Machine Name and Type and returns the Machine ID.

GO

CREATE FUNCTION MachID (@machine varchar(5), @type varchar(10))

RETURNS int AS

BEGIN

DECLARE @machid AS int = 0

SET @machid = (

SELECT machine\_id

FROM machine

JOIN machine\_type ON machine.machine\_type\_id = machine\_type.machine\_type\_id

WHERE machine\_name LIKE @machine

AND @type LIKE machine\_type)

RETURN @machid

END

-- Creating a Stored Procedure for adding new date/shift information. Function returns the identity of newly created row.

GO

CREATE PROCEDURE AddShift (@date AS varchar(10), @shift AS char(1), @time AS varchar(5))

AS

BEGIN

INSERT INTO shift\_date (production\_date, shift\_name\_id, production\_time\_id)

VALUES (

(SELECT CONVERT (date, @date)),

(SELECT shift\_name\_id FROM shift\_name WHERE shift\_name LIKE @shift),

(SELECT production\_time\_id FROM production\_time WHERE production\_time LIKE @time)

)

RETURN @@identity

END

GO

EXEC AddShift '06/13/2019', 'A', 'Day'

SELECT \* FROM shift\_date WHERE shift\_date\_id = @@IDENTITY

-- Creating a Stored Procedure for creating a new product. Function returns the identity of the newly created row.

GO

CREATE PROCEDURE AddProduct (@denier varchar(5), @code varchar(10), @length varchar(5), @type varchar (20))

AS

BEGIN

INSERT INTO product (product\_denier, product\_code, product\_length, product\_type\_id)

VALUES (@denier, @code, @length,

(SELECT product\_type\_id FROM product\_type WHERE product\_type LIKE @type)

)

RETURN @@IDENTITY

END

GO

EXEC AddProduct '18.0', 'PN1163', '2.25', 'Fine Denier'

SELECT \* FROM product WHERE product\_id = @@IDENTITY

GO

-- Creating a Stored Procedure for creating a new customer. Function returns the identity of the newly created row.

CREATE PROCEDURE AddCustomer (@name varchar(40))

AS

BEGIN

INSERT INTO customer (customer\_name)

VALUES (@name)

RETURN @@IDENTITY

END

EXEC AddCustomer 'Mirka'

SELECT \* FROM customer where customer\_id = @@IDENTITY

GO

-- Creating a Stored Procedure for linking a product and customer. Function returns the identity of newly created row.

CREATE PROCEDURE AddProdCust (@denier varchar(5), @code varchar(10), @length varchar(5), @cust varchar(40))

AS

BEGIN

DECLARE @prod AS int = 0,

@custid AS int = 0

SET @prod = (dbo.ProdID(@denier, @code, @length))

SET @custid = (SELECT customer\_id FROM customer WHERE customer\_name LIKE @cust)

INSERT INTO product\_customer\_list (product\_id, customer\_id)

VALUES (@prod, @custid)

RETURN @@IDENTITY

END

EXEC AddProdCust '18.0', 'PN1163', '2.25', 'Mirka'

SELECT \* FROM product\_customer\_list WHERE product\_customer\_id = @@IDENTITY

/\*

Creating a Stored Procedure for inserting new Shift Production data. There are two different types of Production

data - Spinning and Finishing. Though similar, they have different data. For instance, in Spinning, we input the total

amount of pounds spun, as well as Waste and Mix lbs. In Finishing, we input the total amount of good production made, as well

as Waste and Off-Grade. This procedure uses flow control to determine which fields to input and which to leave NULL. Inputs

for the procedure are the date, shift, product code, customer, machine, machine type, total pounds (Spinning or Finishing), (Mix or Off-Grade), Waste,

whether or not product is Excluded. The procedure returns the id for the newly created entry.

\*/

GO

CREATE PROCEDURE AddProdData (@date date, @shift varchar(5), @denier varchar(5), @code varchar(10), @length varchar(5), @customer varchar(40),

@machine varchar(5), @machType varchar(10), @pounds int, @sub int, @waste int, @excluded bit)

AS

BEGIN

DECLARE @dateshift AS int = 0,

@prod AS int=0,

@line AS int = 0

SET @dateshift = dbo.DateID (@date, @shift)

SET @prod = (SELECT product\_customer\_id

FROM product\_customer\_list

JOIN customer ON product\_customer\_list.customer\_id = customer.customer\_id

WHERE product\_id = dbo.ProdID (@denier, @code, @length)

AND customer\_name LIKE @customer)

SET @line = dbo.MachID (@machine, @machType)

IF @machType = 'Spinning'

BEGIN

INSERT INTO shift\_production (shift\_date\_id, product\_customer\_id, machine\_id, spinning\_total\_pounds, mix\_pounds, waste\_pounds, excluded)

VALUES (@dateshift, @prod, @line, @pounds, @sub, @waste, @excluded)

END

ELSE

BEGIN

INSERT INTO shift\_production (shift\_date\_id, product\_customer\_id, machine\_id, finished\_Q1\_pounds, off\_grade\_pounds, waste\_pounds, excluded)

VALUES (@dateshift, @prod, @line, @pounds, @sub, @waste, @excluded)

END

RETURN @@IDENTITY

END

EXEC AddProdData '6/13/2019', 'A', '18.0', 'PN1163', '2.25', 'Mirka', '1', 'Spinning', 10000, 100, 300, 0

SELECT \* FROM shift\_production WHERE shift\_production\_id = @@IDENTITY

EXEC AddProdData '6/13/2019', 'A', '18.0', 'PN1163', '2.25', 'Mirka', '4', 'Finishing', 6000, 100, 240,0

SELECT \* FROM shift\_production WHERE shift\_production\_id = @@IDENTITY

/\*

Creating a function to summarize yield information for a given set of time. The function takes in a start

and end date, and returns a table with the following columns: the sum total of Finished Q1 pounds, Waste, and Sub.

There is another column which returns the total amount of Mix products baled in Finishing. To do this, a subquery

was written b/c this value is ONLY from "Finishing" machines. There is Mix tracked via Spinning, but that amount

does not go into the final yield calculation (this would be double counting, as the Mix from Spinning is processed into

Mix in Finishing.) The function also removes any excluded (R&D) products from the calculation.

This function is a critical first step for calculating Q1 Yield. This function returns a table of the key values which

can be taken by an outside service to run the yield calculation. There is also another created SQL function to return the yield

value.

\*/

GO

CREATE FUNCTION yield\_summary (@start date, @end date)

RETURNS TABLE AS

RETURN (

SELECT SUM(finished\_Q1\_pounds) AS q1,

SUM(off\_grade\_pounds) sub,

SUM(waste\_pounds) waste,

(SELECT SUM(mix\_pounds)

FROM shift\_production

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

JOIN machine ON shift\_production.machine\_id = machine.machine\_id

JOIN machine\_type ON machine.machine\_type\_id = machine\_type.machine\_type\_id

WHERE production\_date BETWEEN @start AND @end

AND excluded = 0

AND machine\_type LIKE 'Finishing') AS mix

FROM shift\_production

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

WHERE production\_date BETWEEN @start AND @end

AND excluded = 0

)

SELECT q1 FROM dbo.yield\_summary ('5/1/2019', '5/8/2019')

/\*

Creating a function to return the Q1% for a given time period. This is the most basic yield value we use, and while most likely

these calculations should be done by an external software retrieving the data from the yield\_summary function, I wanted to provide

a function for doing it within SQL. This function takes in 2 dates, runs the yield\_summary function to get the summarized yield data

and then returns a calculated Q1% value. This value is (using column names from the yield\_summary function)

Q1% = (q1 / (waste + sub + mix + q1)) \* 100

One complication when initially creating this function is that, since all the values are integers, the calculation would return only 0. To

beat this, I cast the values as floats for the calculation and return a decimal value (with 2 decimal places of precision)

\*/

GO

CREATE FUNCTION q1\_calc (@start date, @end date)

RETURNS decimal(5,2) AS

BEGIN

DECLARE @q1 AS decimal(5,2)

SET @q1 = (SELECT CAST(q1 AS float) / CAST(q1 + mix + waste + sub AS float) \* 100

FROM dbo.yield\_summary(@start,@end))

RETURN @q1

END

SELECT dbo.q1\_calc('5/1/2019', '5/2/2019')

-- Quick function for calculating a % of the total given the appropriate values

GO

CREATE FUNCTION percentcalc (@num int, @denom1 int, @denom2 int, @denom3 int)

RETURNS decimal (5,2) AS

BEGIN

DECLARE @q AS decimal(5,2)

SET @q = CAST(@num AS float) / CAST(@num + @denom1 + @denom2 + @denom3 AS float) \* 100

RETURN @q

END

SELECT q1, sub, waste, mix,

dbo.percentcalc(waste, sub, q1, mix)

FROM dbo.yield\_summary('5/1/2019', '5/31/2019')

/\*

Creating a stored procedure to update the Waste for a given shift\_production field.

The procedure takes in the date and shift for the update, as well as the product/customer

and machine. It also takes in the value (in pounds) of waste to be added to the specific field.

The waste is added to the current value in waste, not simply replacing the value in there.

\*/

GO

CREATE PROCEDURE AddWaste (@date date, @shift varchar(5), @denier varchar(5), @code varchar(10), @length varchar(5), @customer varchar(40),

@machine varchar(5), @machType varchar(10), @waste int)

AS

BEGIN

DECLARE @dateshift AS int = 0,

@prod AS int=0,

@line AS int = 0

SET @dateshift = dbo.DateID (@date, @shift)

SET @prod = (SELECT product\_customer\_id

FROM product\_customer\_list

JOIN customer ON product\_customer\_list.customer\_id = customer.customer\_id

WHERE product\_id = dbo.ProdID (@denier, @code, @length)

AND customer\_name LIKE @customer)

SET @line = dbo.MachID (@machine, @machType)

UPDATE shift\_production

SET waste\_pounds = waste\_pounds + @waste

WHERE shift\_date\_id = @dateshift

AND product\_customer\_id = @prod

END

EXEC AddWaste'6/13/2019', 'A', '18.0', 'PN1163', '2.25', 'Mirka', '1', 'Spinning', 100

# Answering Questions with the Data

Here we use the data we are collecting to answer the basic questions we posed earlier in this report. The question is posed again, and the view created to answer the question is underneath, along with example screenshots of what the view returns.

1. What is the plant performance for a given month/year?

For this question, two views were created, one summarizing yield performance Year-To-Date, and another summarizing yield performance Month-To-Date.

/\*

The first data question - and the most commonly accessed - is a look at plant performance for

a given time period - typically Year-To-Date and Month-To-Date. We want to know what the total

pounds of Q1 product, Waste, Mix, and off-grade; as well as the % of the total each is. We will

create 2 views for this - one for YTD and one for MTD. As stated above in the Q1 calculation function,

this view sums all Finishing, Waste, and Off-Grade totals, but only totals the Mix from Finishing, in order

to avoid double-counting.

\*/

GO

CREATE VIEW YTDView AS

SELECT q1 AS 'Q1 Pounds',

sub AS 'Off-Grade Pounds',

waste AS 'Waste Pounds',

mix AS 'Mix Pounds',

dbo.q1\_calc(DATEFROMPARTS(DATEPART(year, GETDATE()),1,1), GETDATE()) AS 'Q1 %',

cast(cast(waste AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Waste %',

cast(cast(sub AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Off-Grade %',

cast(cast(mix AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Mix %'

FROM dbo.yield\_summary(DATEFROMPARTS(DATEPART(year, GETDATE()),1,1), GETDATE())

CREATE VIEW MTDView AS

SELECT q1 AS 'Q1 Pounds',

sub AS 'Off-Grade Pounds',

waste AS 'Waste Pounds',

mix AS 'Mix Pounds',

dbo.q1\_calc(DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month, GETDATE()),1), GETDATE()) AS 'Q1 %',

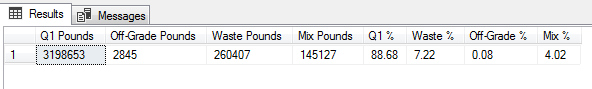
cast(cast(waste AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Waste %',

cast(cast(sub AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Off-Grade %',

cast(cast(mix AS float) / cast(q1 + sub + waste + mix AS float) \*100 AS decimal(5,2)) AS 'Mix %'

FROM dbo.yield\_summary(DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month,GETDATE()),1), GETDATE())

The views above return summaries of the total pounds produced in the time frame, and a % of the total for each of the categories. Below is an example of the Year-To-Date summary.



1. What is the Yield for a specific Machine/Machine Type for a given time period or shift

Similar to above, we often want to know what the specific yields are for a certain Production Machine. Here, there was a bit of a difficult situation because of the way data is collected. In the Spinning process, we measure Input pounds, and so the amount of good finished pounds is calculated as the total less any Waste or Mix. In Finishing, however, it’s the opposite. We measure the output good pounds, as well as the various off-quality amounts, and use the sum total of all to determine the input amount.

I wanted to create a view showing all production machines, but had to deal with this difference in calculation. To do it, I ended up using CASE statements, which used different equations depending on the Machine Type.

Two different views were created – one for Month-To-Date and one for Year-To-Date. The code is below and a representative sample from the Month-To-Date view follows.

/\* The second data question is what are the specific yields (pounds and %'s) for a given Machine/Machine Type for a given

time period. Again, this information is often looked at on a MTD and YTD basis. The created view sums the total pounds for the specific

machine and also calculates %'s of the total for each category. There is one view for MTD and one view for YTD.

Because of the way production pounds are measured, there are 2 different methods of calculating the total percents. In Spinning, we measure

the total amount of production that was produced (ie - INPUT pounds), and from that we must subtract the amount of Waste and Mix created to find

the Q1 %. In Finishing, we measure the total amount of good pounds produced (ie - OUTPUT pounds) along with the amount of Waste, Mix, and Off-Grade

produced. In order to display the correct % values for both types in the same view, I used several CASE statements - with one equation if calculating

for Spinning and one statement if calculating for Finishing.

\*/

GO

CREATE VIEW MachineYTD AS

SELECT machine\_type + ' ' + machine\_name AS 'Machine Name',

SUM(spinning\_total\_pounds) AS 'Total Spun Pounds' ,

SUM(finished\_Q1\_pounds) AS 'Finished Q1 Pounds',

SUM(waste\_pounds) AS 'Waste Pounds',

SUM(mix\_pounds) AS 'Mix Pounds',

SUM(off\_grade\_pounds) AS 'Off-Grade Pounds',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(spinning\_total\_pounds) - SUM(waste\_pounds) - SUM(mix\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(finished\_q1\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Q1 %',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(waste\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(waste\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Waste %',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(mix\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(mix\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Mix %',

cast(cast(SUM(off\_grade\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2)) AS 'Sub %'

FROM shift\_production

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

JOIN machine ON shift\_production.machine\_id = machine.machine\_id

JOIN machine\_type ON machine.machine\_type\_id = machine\_type.machine\_type\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),1,1) AND GETDATE()

GROUP BY machine\_type, machine\_name

GO

CREATE VIEW MachineMTD AS

SELECT machine\_type + ' ' + machine\_name AS 'Machine Name',

SUM(spinning\_total\_pounds) AS 'Total Spun Pounds' ,

SUM(finished\_Q1\_pounds) AS 'Finished Q1 Pounds',

SUM(waste\_pounds) AS 'Waste Pounds',

SUM(mix\_pounds) AS 'Mix Pounds',

SUM(off\_grade\_pounds) AS 'Off-Grade Pounds',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(spinning\_total\_pounds) - SUM(waste\_pounds) - SUM(mix\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(finished\_q1\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Q1 %',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(waste\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(waste\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Waste %',

CASE

WHEN machine\_type LIKE 'Spinning' THEN cast(cast(SUM(mix\_pounds) AS float)/cast(SUM(spinning\_total\_pounds) AS float) \* 100 AS decimal(5,2))

WHEN machine\_type LIKE 'Finishing' THEN cast(cast(SUM(mix\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2))

END AS 'Mix %',

cast(cast(SUM(off\_grade\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(mix\_pounds) + SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2)) AS 'Sub %'

FROM shift\_production

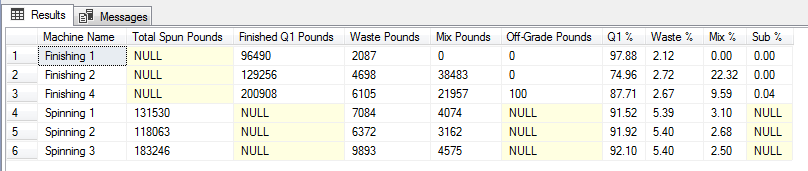
JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

JOIN machine ON shift\_production.machine\_id = machine.machine\_id

JOIN machine\_type ON machine.machine\_type\_id = machine\_type.machine\_type\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month, GETDATE()),1) AND GETDATE()

GROUP BY machine\_type, machine\_name



1. What are our best/worst performing products?

For this question, I created 2 views, one giving the Top 10 products based on their year-to-date Q1% performance. The other view lists the Bottom 10 products in the same list. For the second list, I also removed any products from the list that had no production.

/\*

One question we look at regularly is what are our Best and Worst performing products. Two views are created which look at the Top 10

and Bottom 10 products in terms of Q1% YTD.

\*/

GO

CREATE VIEW Bottom10 AS

SELECT TOP 10

product\_denier + '-' + product\_code + '-' + product\_length AS 'Product',

CAST(CAST(SUM(finished\_q1\_pounds) AS float)/CAST(SUM(finished\_q1\_pounds)+SUM(waste\_pounds)+SUM(mix\_pounds)+SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2)) AS 'Q1%'

FROM shift\_production

JOIN product\_customer\_list ON shift\_production.product\_customer\_id = product\_customer\_list.product\_customer\_id

JOIN product ON product\_customer\_list.product\_id = product.product\_id

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),1,1) AND GETDATE()

GROUP BY product\_denier, product\_code, product\_length

HAVING SUM(finished\_q1\_pounds) | SUM(waste\_pounds) | SUM(mix\_pounds) | SUM(off\_grade\_pounds) IS NOT NULL

ORDER BY [Q1%] ASC

GO

CREATE VIEW Top10 AS

SELECT TOP 10

product\_denier + '-' + product\_code + '-' + product\_length AS 'Product',

CAST(CAST(SUM(finished\_q1\_pounds) AS float)/CAST(SUM(finished\_q1\_pounds)+SUM(waste\_pounds)+SUM(mix\_pounds)+SUM(off\_grade\_pounds) AS float)\*100 AS decimal(5,2)) AS 'Q1%'

FROM shift\_production

JOIN product\_customer\_list ON shift\_production.product\_customer\_id = product\_customer\_list.product\_customer\_id

JOIN product ON product\_customer\_list.product\_id = product.product\_id

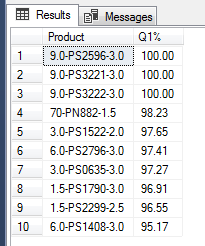
JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

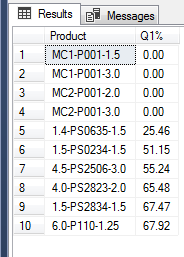
WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),1,1) AND GETDATE()

GROUP BY product\_denier, product\_code, product\_length

HAVING SUM(finished\_q1\_pounds) | SUM(waste\_pounds) | SUM(mix\_pounds) | SUM(off\_grade\_pounds) IS NOT NULL

ORDER BY [Q1%] DESC





1. Who are our most valuable customers (in terms of production volume..)?

To answer this question, I created a view that lists the Top 10 customers based on volume of products made for them Year-To-Date. The View returns the Customer name and how many pounds we have produced for them this year.

/\*

The next data question this database will allow us to answer is - who are our most valuable customers (in terms of volume)? This view looks at

the Top 10 Customers and their volumes YTD.

\*/

GO

CREATE VIEW TopCustomers AS

SELECT TOP 10

customer\_name AS 'Customer Name',

SUM(finished\_q1\_pounds) AS 'Total Volume'

FROM shift\_production

JOIN product\_customer\_list ON shift\_production.product\_customer\_id = product\_customer\_list.product\_customer\_id

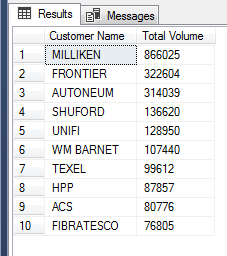
JOIN customer ON product\_customer\_list.customer\_id = customer.customer\_id

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),1,1) AND GETDATE()

GROUP BY customer\_name

ORDER BY [Total Volume] DESC



1. How many products are run during a given time period or shift?

Another relatively simple question, but one that we look at often. I created a view that counts the number of distinct products run Month-To-Date. Often we want to know how many unique products we ran, but don’t discriminate if we ran the same product for 2 different customers. So this view returns a count of distinct products run during the month.

/\*

We also often want to know how many unique products we ran in a given month. This view returns the count of distinct

products (not product-customer pairs, just distinct prodcuts) run MTD.

\*/

GO

CREATE VIEW MonthlyProductCount AS

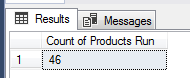
SELECT COUNT (DISTINCT product\_id) AS 'Count of Products Run'

FROM shift\_production

JOIN product\_customer\_list ON shift\_production.product\_customer\_id = product\_customer\_list.product\_customer\_id

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month, GETDATE()),1) AND GETDATE()



1. What is the Yield for a specific Product Type for a given time period or shift?

Finally, the last data question seeks to know what the Q1% is for our two different product types – Fine and Heavy Deniers. A view was created which returns the total number of pounds and Q1% for each product type.

As before, when we looked at the yields by Machine Type, we have differences here between types that require different equations. Heavy Denier fibers are one-stage processes, while Fine Denier are 2-stage processes. That is, Heavy Denier products are only run on Finishing lines, but Fine Denier requires both Spinning and Finishing (and all products on Spinning lines are Fine Denier). So, to account for the Mix like we did above, and separate the 2 types, I used another CASE statement for the 2 different equations.

/\*

The final question we want to answer (here at least) is what is the Production Q1% Yield for a given

Product Type. That is - What is the Yield for Fine Denier and Heavy Denier products in a given time period -

in this case, MTD.

As before in our Q1 calculations, we have to account for Mix showing up in both the Spinning stages and the Finishing stages.

An additional wrinkle emerges here - The "Heavy Denier" products are a 1-stage product - meaning they ONLY go through

Finishing (and do not generate Mix). All products run on the "Spinning" step are "Fine Denier" products, and so all Mix must be attributed

only to them. To solve this problem, I created another CASE statement, with 2 different equations for Q1% depending on whether the product

is Fine Denier or Heavy Denier.

\*/

GO

CREATE VIEW ProductTypeYield AS

SELECT product\_type AS 'Product Type',

SUM(finished\_Q1\_pounds) AS 'Total Pounds',

CASE

WHEN product\_type LIKE 'Fine Denier'

THEN cast(cast(SUM(finished\_q1\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(off\_grade\_pounds) +

(SELECT SUM(mix\_pounds)

FROM shift\_production

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

JOIN machine ON shift\_production.machine\_id = machine.machine\_id

JOIN machine\_type ON machine.machine\_type\_id = machine\_type.machine\_type\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month, GETDATE()),1) AND GETDATE()

AND machine\_type LIKE 'Finishing') AS float) \* 100 AS decimal(5,2))

WHEN product\_type LIKE 'Heavy Denier'

THEN cast(cast(SUM(finished\_q1\_pounds) AS float)/cast(SUM(finished\_q1\_pounds) + SUM(waste\_pounds) + SUM(off\_grade\_pounds) AS float) \* 100 AS decimal(5,2))

END AS 'Q1%'

FROM shift\_production

JOIN product\_customer\_list ON shift\_production.product\_customer\_id = product\_customer\_list.product\_customer\_id

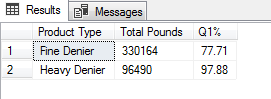
JOIN product ON product\_customer\_list.product\_id = product.product\_id

JOIN product\_type ON product.product\_type\_id = product\_type.product\_type\_id

JOIN shift\_date ON shift\_production.shift\_date\_id = shift\_date.shift\_date\_id

WHERE production\_date BETWEEN DATEFROMPARTS(DATEPART(year, GETDATE()),DATEPART(month, GETDATE()),1) AND GETDATE()

GROUP BY product\_type

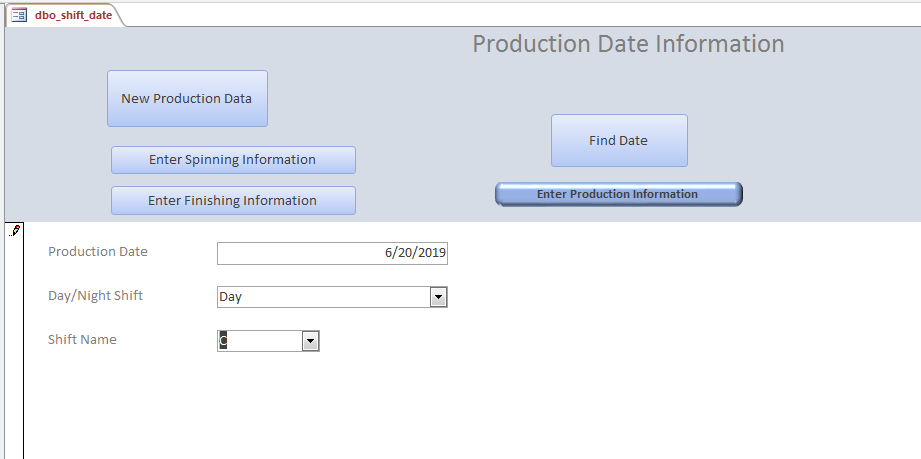


# User Interface

For users, I wanted an interface where shift production data could be easily input. This is by far the biggest interaction one will have with the database. It is critical that the data – the Date, Shift, Product, and all measurements are input correctly. We have issues with typos corrupting our data. With 6 Supervisors inputting data at the end of their shifts, mistakes are often made. I want something robust enough to prevent as many of these errors as possible.

Ideally, the interface I would like to use for inputting data is a Web Form using PHP. But first, I need to learn how to code in PHP! In the meantime, I’ve mocked up forms similar to what I want through Access.

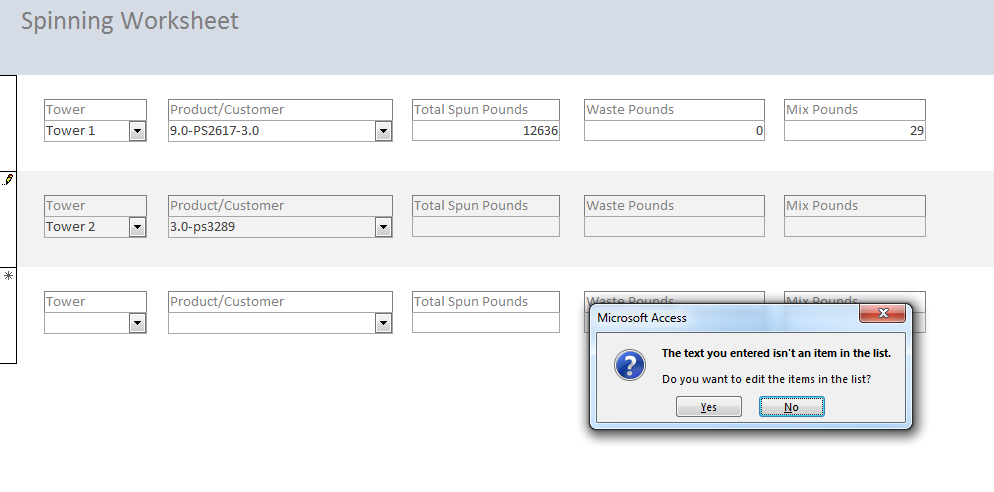
The first screen I’ve designed for data entry is where a user can input the date, the shift, and whether it is Day or Night shift. From there, they can select to input either Spinning or Finishing data.



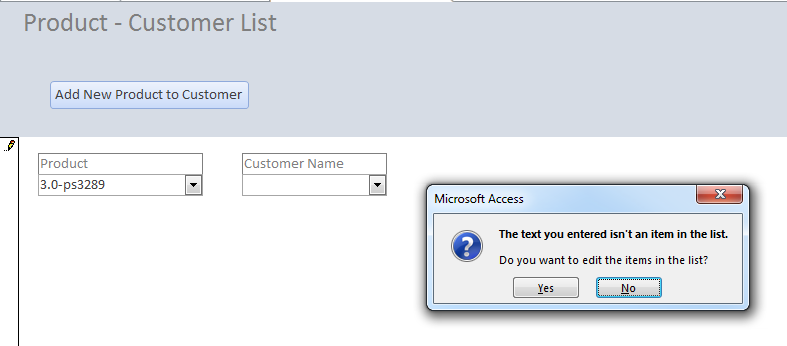
Pressing the Enter Spinning Information button brings you to a new page where the Spinning shift-production data is entered. The date and shift selected on the previous screen is brought in as the default here.

On the Spinning page (and similarly with the Finishing page), there is a combo box to select the correct Production line. Then there is another combo box to add the correct Product and Customer. Then the user inputs the Total Spinning pounds, Mix, and Waste. All other fields are defaulted behind the scenes to NULL.

If the product-customer combination is not in the list, the following error is given. Selecting Yes redirects the user to another form to create a new product-customer link.



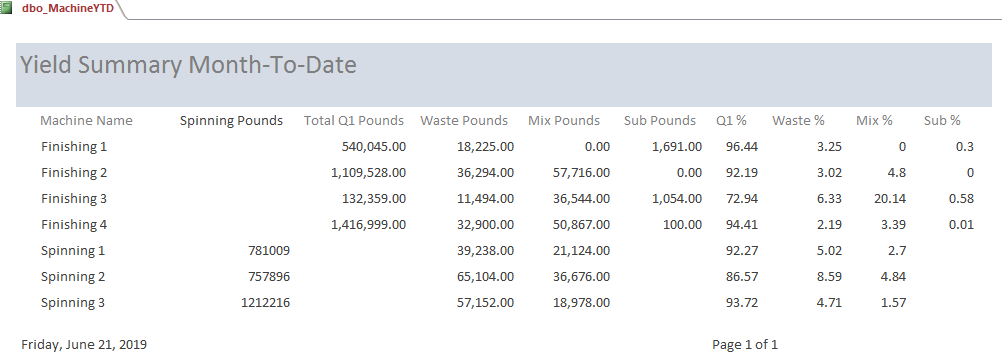
The redirected screen has a button allowing a user to create a new Product-Customer link. Again, there are two combo-boxes for Products and Customer. If you input a product that does not exist, another prompt comes up to add new product information. The same thing if a new customer is inputted.



The New Product screen is where the Product’s Denier, Code, and Length can be inputted, as well as whether it is Fine or Heavy Denier. Once created, closing the window brings them back to the Product-Customer screen where they can continue creating new links, and back to the production entry page.



As far as reporting, for now at least, most reporting is done by me, with data in Excel used to make various tables and charts. Eventually, I plan to use R and other data visualization to utilize the views created in SQL, but for now, several reports have been created just to show the data provided by the view. As an example, here is the report for the Month-To-Date Yield Summary by Machine



# Reflections

This database became way more difficult than I’d initially expected it to be, but I see the value in it, and am excited to be able to roll it out soon for actual use. When I first considered what the database might look like, I only thought it would be a couple of tables. But as I went through the normalization process, and considered what I would need in order to protect data integrity, the number of tables began to balloon. I also expected not to have difficulty in manipulating the data to get the answers I needed from the data. But as I started to pull the data, and realized how our complicated business rules have made complicated data, I began to have trouble in creating accurate queries to get what I needed.

These complications stretched my abilities, and helped me learn new ways to think about the problem. I found myself often thinking how might I better get the data I needed. Though my database doesn’t have much data in it right now, I know how quickly it will expand – with thousands of rows of data created annually. So I found myself thinking how to best optimize my queries with that future in mind.