Revenue Improvement Database

Database project for 0406 - IST659

Scott Snow

6/29/2018

Table of Contents

Project Deliverable 2

Summary

This database overall serves to increase the profitability of the company I work for which is Premier Building Systems. We manufacture building panels called SIP's which stand for *Structurally Insulated Panels* and this is under our brand Premier SIPS. The primary stakeholders are myself and my production manager. Secondary stakeholders include our *lead operators*. A glossary is included at the end.

Our production team is split into four groups: Shift A and Shift B alternate schedules to keep two of three production lines running seven days a week. Shift C works ten hour days five days a week. For the purposes of this database the management and office staff are grouped together in the fourth category. On a given day, each employee operates in six production areas: Production Lines 1 and 2, *Router*, *Spline/Header*, Laminator, and Office. We also have shipping and maintenance departments which are outside the scope of this database.

Overall we seek to track various dollar amounts including revenue and cost as well as time and rate amounts such as panels per hour and job completion time. Having recently moved to a new building and being under new ownership, this allows us the opportunity to reevaluate many of our policies and processes for efficiency. Having discussed this opportunity with my production manager, these are five questions that we would like to learn the answer too.

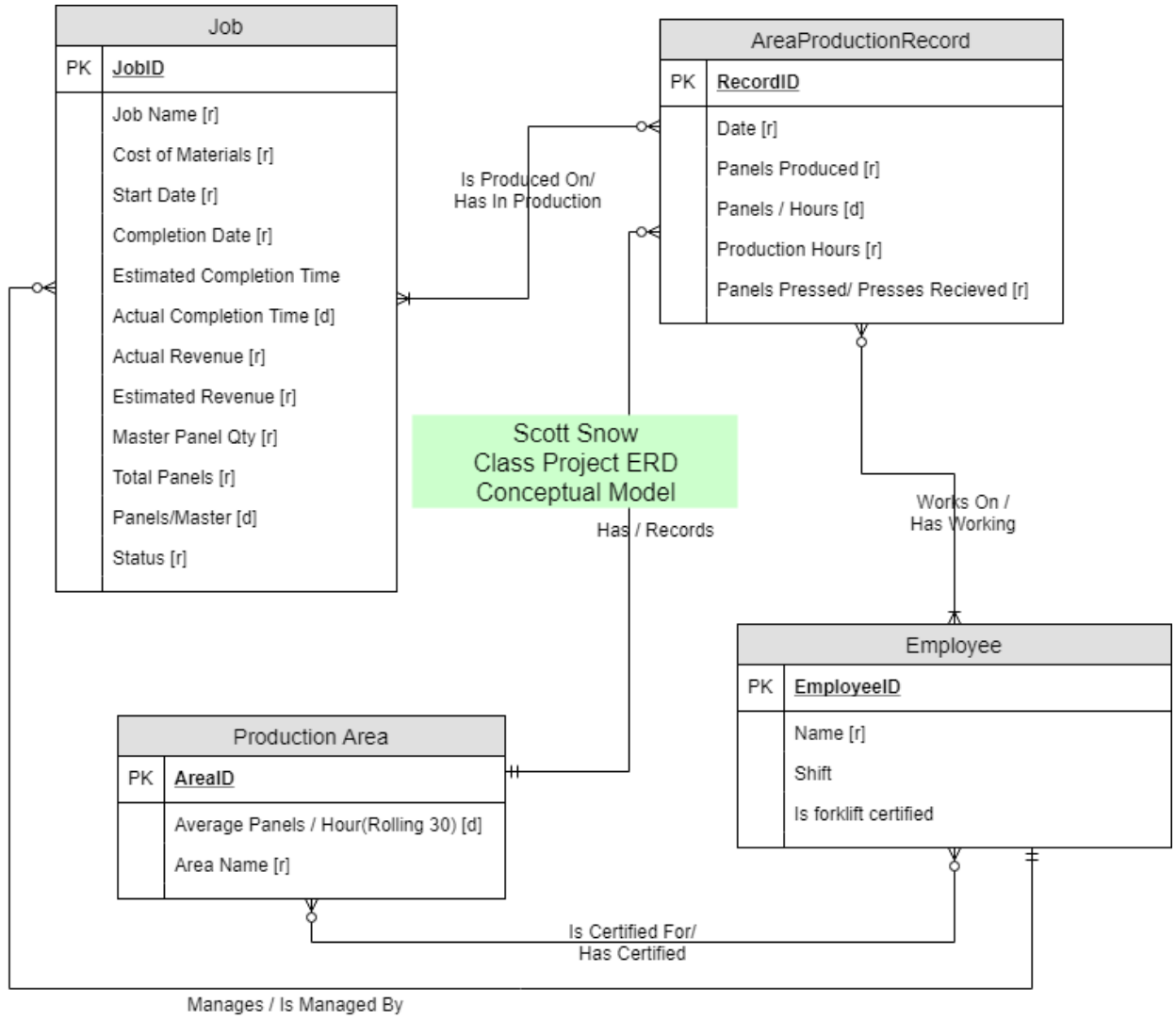
1. Is production time, material cost, or overhead cost more correlated with a project's revenue?
    My manager has noticed an inconsistency with project revenue and the time it takes to finish. He details instances were some jobs that generated a certain amount of revenue took multiple days to complete while other jobs that generated twice as much revenue were completed in much less time. After discussing what goes into determining how much a customer is charged, the following questions build on this first question to determine the need for potential changes.
2. There should be a strong positive correlation between a project's revenue and the identified factor in question 1. Is this the case?
    This is to determine whether what my manager describes can be explained by other means.
3. Does the data suggest that we should make changes to our pricing structure and negotiations?
    This is a culmination of the first two questions and is ultimately one of the most important aspects of improving our profitability.
4. Is the time spent doing *non-price increasing but still customer added activities* at a reasonable amount.
    There are many things that we do to a product that are included in the base price of a panel. The issue is that if we are spending too much production time on these activities, it slows down the other sections of a production area that are generating more value by being faster. While the best way to measure this would be to take

timestamp data of a panel's process through the production area, this is not feasible at this time. As an alternative, we are looking to derive the average panel production time of *stock panels* versus fabricated panels in which the time to produce stock panels should be a small percentage of the time to produce fabricated panels. These values are derived from production values

5. Would the press operation benefit from an additional smaller laminator and batch press machine?

   While this is a smaller concern for my manager, we receive many requests for our *InsulLam* product. With our current machinery, we can press up to four of these in an 8 minute span. While this new machinery would sharply increase the amount of these that we are able to press at a time, it is unclear whether this new machinery, which is costly, would actually go to significant use. There is a limited staging area for each production area and if it is full, no further panels can be added to that line.

Conceptual Model



In our entity relation diagram, we have four entities. There are employees that are certified for various production areas and work in a various production area on a given day. For various reasons these are not always the same. Additionally, there are project managers among our employees as well as lead operators for each shift. Also each employee is either certified to operate a forklift, or they are not. While this is usually a prerequisite for an area certification, it is not absolute and an employee can be forklift certified without needing to be certified for an area.

There are production areas that have various operators certified to lead in those areas. Additionally, if an area is in production on a given day, then it will have an associated production area record for that day.

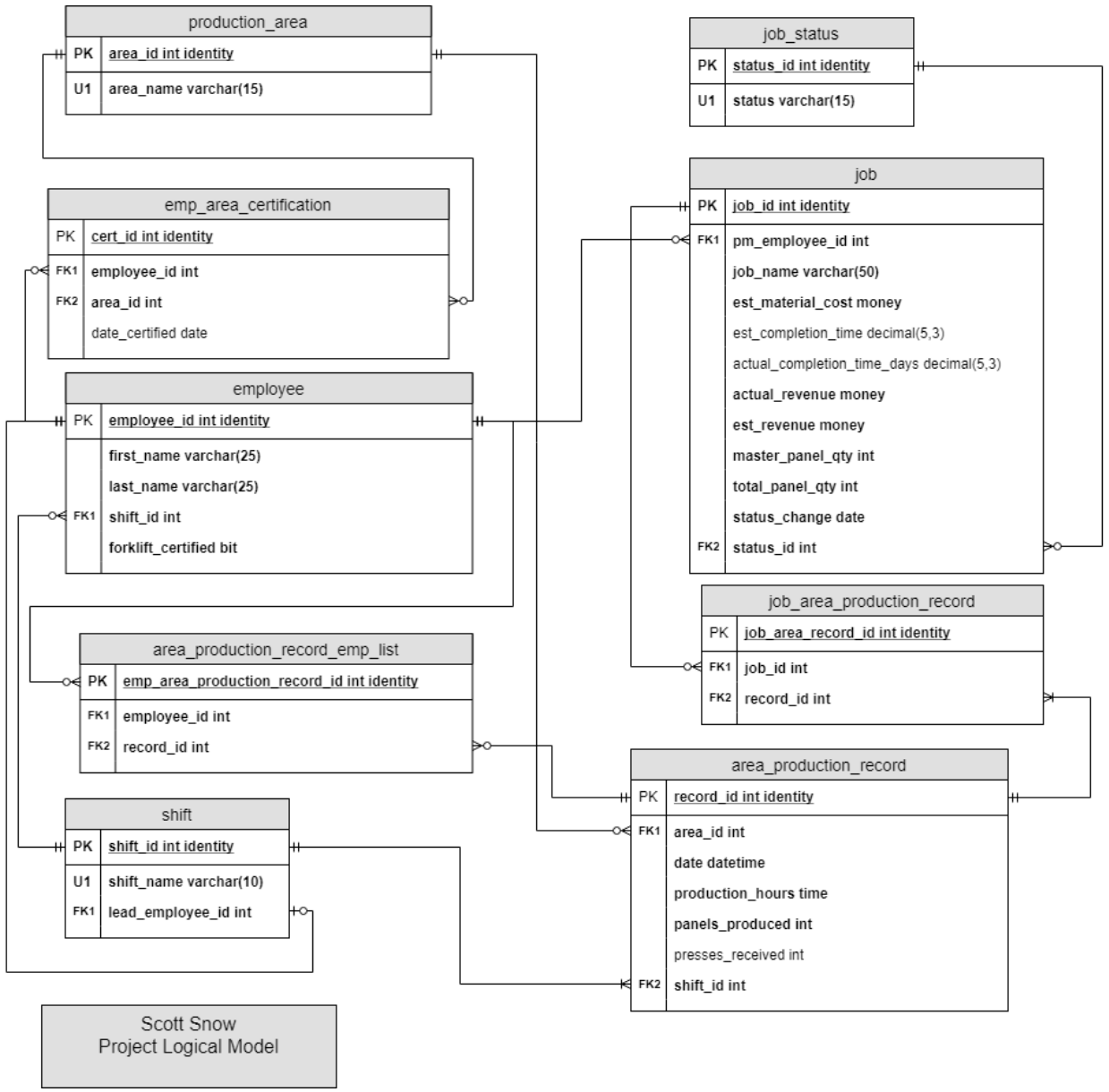
Each job is managed by a project manager. Each job has a production status. The four production statuses are pre-production, in progress, post-production, completed. When a job is in progress, it will appear on each new production area records until it enters post-production or is put on hold. Each job contains main required materials for the panels as well as additional optional materials such as lumber, splines, and headers. Each panel is grouped with others into a master panel which is pressed in the lamination area.

Each production area record exists for exactly one day and exactly one area. It documents the employees and the shift that worked in that area. It documents the output of that area for a specific day. It details the Jobs that were worked in that area for that day. Areas like the laminator and spline/header work multiple jobs in a day. The other areas sometimes work one or sometimes work multiple jobs depending on how many jobs they complete. For a record to exist for an area, it has to have worked on at least one job that day. Some areas are not in production for the full day so production hours are included as an attribute.

As time moved forward, we took out the estimated fields for the Job table as they weren't of much use. Additionally, the field of Final Overhead Cost as added during physical design. Finally, Average Panels / Hour in Production Area and Panels/Master in Job were concluded to be unnecessary fields so they were removed.

Logical Model

In transferring the conceptual model to a logical model, I introduced some newer entities that constrained some of the attributes from the conceptual model. This new entities are shift and status. Aside from the conceptual entities which became relations, I had three many-to-many relationships which required bridge tables. Most of the data types are self explanatory. One that was notable was bit for employee: forklift_certified. This is because that attribute can have only two potential values so it makes sense.



During implementation, we discovered that we did not have enough information to correctly answer the fifth question. Therefore, we included a 10th table that lists every station in the production and combines with the area that each station is in to form a unique constraint. Again, the estimated value fields for Job were removed. Additionally, we added production hours to Job-Area Production Record to allow us to better determine production hours as opposed to taking the difference between start and finish dates as planned.

Glossary

Structurally Insulated Panels –This is our main product. It is two sheets of oriented strand board laminated together with a structural expanded polystyrene foam core.

Lead operator – Each employee that works in production under our production manager is considered an operator. A lead operator leads a shift and a certified operator leads a production area.

Router – Our router is a large CNC machine that can fabricate panels.

Spline – Spline is a product we make that that is essentially a narrow 3 inch structural panel that is used in the place of lumber.

Header – Header is a product we make that is structural LVL lumber with a foam core between two pieces. It is used in place of wood beams in home construction.

Non-price increasing but still customer added activities – This includes our fabrication processes such as recessing the panel for lumber installation, sealing the edges of the strand board from moisture and labeling the panel for the customer.

Stock panels – These panels are taken directly from the press and given to customers who wish to complete their own fabrication.

InsulLam – This is a product used just for insulation. It is one sheet of strand board or another wood product laminated together with a foam core.

QC – Quality Control

Hogger – The hogger is an air powered tool that spins three blades rapidly to cut foam. A frame sits on the edge of a panel to recess the panel to a certified depth making room for lumber to be installed.

Table Creation Code

```
/* Scott Snow
	IST 659
	April 2018 Term
	Course Project Table Creation
	Premier Building Systems Operations Database
*/
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME =
'pbs_AreaProductionRecordEmployeeList')
BEGIN
	DROP TABLE pbs_AreaProductionRecordEmployeeList
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME =
'pbs_JobAreaProductionRecord')
BEGIN
	DROP TABLE pbs_JobAreaProductionRecord
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Station')
BEGIN
	DROP TABLE pbs_Station
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME =
'pbs_AreaProductionRecord')
BEGIN
	DROP TABLE pbs_AreaProductionRecord
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME =
'pbs_AreaCertification')
BEGIN
	DROP TABLE pbs_AreaCertification
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Job')
BEGIN
	DROP TABLE pbs_Job
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Area')
BEGIN
	DROP TABLE pbs_Area
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Shift')
BEGIN
	ALTER TABLE pbs_Shift
		DROP CONSTRAINT IF EXISTS FK_Lead
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Employee')
BEGIN
	ALTER TABLE pbs_Employee
```

```
            DROP CONSTRAINT IF EXISTS FK_Shift
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Employee')
BEGIN
	DROP TABLE pbs_Employee
END
GO

IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_Shift')
BEGIN
	DROP TABLE pbs_Shift
END
GO
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'pbs_JobStatus')
BEGIN
	DROP TABLE pbs_JobStatus
END
GO

CREATE TABLE pbs_Area (
	AreaID int identity primary key
	, AreaName char(15) UNIQUE not null
)
GO

CREATE TABLE pbs_Shift (
	ShiftID int identity primary key
	, ShiftName char(10) UNIQUE not null
	, LeadOperatorID int
)
GO

CREATE TABLE pbs_Employee (
	EmployeeID int identity primary key
	, FirstName varchar(25) not null
	, LastName varchar(25) not null
	, ShiftID int  not null
	, isForkliftCertified bit not null
	CONSTRAINT FK_Shift FOREIGN KEY(ShiftID) REFERENCES pbs_Shift(ShiftID)
	CONSTRAINT uc_Names UNIQUE(FirstName, LastName)
)
GO

CREATE TABLE pbs_AreaCertification (
	CertID int identity primary key
	, EmployeeID int FOREIGN KEY REFERENCES pbs_Employee(EmployeeID) not null
	, AreaID int FOREIGN KEY REFERENCES pbs_Area(AreaID) not null
	, DateCertified datetime
)
GO

CREATE TABLE pbs_AreaProductionRecord (
	RecordID int identity primary key
	, AreaID int FOREIGN KEY REFERENCES pbs_Area(AreaID) not null
	, ProductionDate date not null
	, ProductionHours decimal(4,2) not null
	, Panels int not null
```

```
	, PressesIO int
	, ShiftID int FOREIGN KEY REFERENCES pbs_Shift(ShiftID) not null
)
GO


CREATE TABLE pbs_JobStatus (
	StatusID int identity primary key
	, StatusText varchar(15) UNIQUE not null
)
GO

CREATE TABLE pbs_Job (
	JobID int identity primary key
	, JobName varchar(50) not null
	, MasterPanelQty int not null
	, PanelQty int not null
	, StatusChangeDate datetime DEFAULT GETDATE() not null
	, ProjectManagerID int FOREIGN KEY REFERENCES pbs_Employee(EmployeeID)
	, StatusID int FOREIGN KEY REFERENCES pbs_JobStatus(StatusID)
	, ProjectNumber varchar(40)
	, FinalRevenue decimal(11,2)
	, FinalMaterialCost decimal(11,2)
	, FinalOverhead decimal(11,2)
)
GO

CREATE TABLE pbs_JobAreaProductionRecord (
	JobAreaRecordID int identity primary key
	, JobID int FOREIGN KEY REFERENCES pbs_Job(JobID) not null
	, RecordID int FOREIGN KEY REFERENCES pbs_AreaProductionRecord(RecordID) not null
	, TotalHours decimal (4,2) not null
)
GO

CREATE TABLE pbs_Station (
	StationID int identity primary key
	, StationName varchar(30) not null
	, AreaID int FOREIGN KEY REFERENCES pbs_Area(AreaID) not null
	, CONSTRAINT pbs_St UNIQUE(StationName, AreaID)
)
GO

CREATE TABLE pbs_AreaProductionRecordEmployeeList (
	EmpAreaProdRecordID int identity primary key
	, EmployeeID int FOREIGN KEY REFERENCES pbs_Employee(EmployeeID) not null
	, RecordID int FOREIGN KEY REFERENCES pbs_AreaProductionRecord(RecordID) not null
	, StationID int FOREIGN KEY REFERENCES pbs_Station(StationID) not null
)
GO
```

Constant Data Insert

This query consisted of inserting data that is only going to change if we remove an entire process or add some new machine. The last time this would have changed was when we added our second fabrication line during the winter of 2016-2017.

```
/* Scott Snow
	IST 659
	Course Project
	Premier Building Systems: Operations Database
	Inserting constant data
		Area
		Shift
		JobStatus
		Station

*/
-- Create Shifts
INSERT INTO pbs_Shift (ShiftName)
VALUES
	('A Shift'),
	('B Shift'),
	('C Shift'),
	('Office')



-- Create Areas
INSERT INTO pbs_Area (AreaName)
VALUES
	('Lamination'),
	('Splines'),
	('Fabrication 1'),
	('Fabrication 2'),
	('Router'),
	('Office')


-- Create Area Stations

INSERT INTO pbs_Station (StationName, AreaID)
VALUES ('QC', 4), ('Foam Leader', 1), ('Foam Cutter', 1), ('Laminator', 1), ('Press', 1),
('Hoist', 1), ('Press Operator 4', 1), ('Press Operator 5', 1),
('Rail Saw', 1),('Rail Saw Assistant', 1),('Splines Lead', 2),('Splines Assistant',
2),('Layout', 3),('Floater', 3), ('Cutter', 3),('Hogger', 3),
('Scrape and Edge Seal 1', 3), ('Scrape and Edge Seal 2', 3),('Lumber Install', 3),
('Lumber Assistant', 3), ('Lumber Cutter', 2), ('Router Operator', 5),
('QC', 3), ('Insulam Fab 1', 1), ('Insulam Fab 2', 1), ('Cutter', 4),('Hogger', 4),
('Scrape and Edge Seal 1', 4), ('Scrape and Edge Seal 2', 4),('Lumber Install', 4),
('Lumber Assistant', 4),
('Hogger', 5), ('Scrape and Edge Seal 1', 5), ('Scrape and Edge Seal 2', 5),('Lumber
Install', 5), ('Lumber Assistant', 5), ('CA Stickers', 3), ('CA Stickers', 4), ('CA
Stickers', 5),
('Foam Leader 2', 1), ('Foam Cutter 2', 1), ('Layout', 4), ('Floater', 4)
```

```
-- Create job statuses
INSERT INTO pbs_JobStatus (StatusText)
VALUES ('Pre-Production'), ('Started'), ('In Progress'), ('On Hold'), ('Post-
Production'), ('Completed')
```

Initial Data Insert

This query consisted of inserting semi constant data such as employee's and area certifications. It also consisted of inserting some initial production data that had already been collected.

```
/* Inserting initial Data as of the database creation
This includes Employees, Jobs, and Area Production Records
*/


-- Create Shift Leads
INSERT INTO pbs_Employee (FirstName, LastName, ShiftID, isForkliftCertified)
VALUES
	('Alexander', 'Holaday', 2, 1),
	('Sarek', 'Chhthoun', 1, 1),
	('Dan', 'Glassman', 3, 0),
	('Russ', 'Highland', 4, 0)

-- Set Shift Leads
UPDATE pbs_Shift
SET pbs_Shift.LeadOperatorID = 1
WHERE ShiftID = 2
UPDATE pbs_Shift
SET pbs_Shift.LeadOperatorID = 2
WHERE ShiftID = 1
UPDATE pbs_Shift
SET pbs_Shift.LeadOperatorID = 3
WHERE ShiftID = 3
UPDATE pbs_Shift
SET pbs_Shift.LeadOperatorID = 4
WHERE ShiftID = 4

-- Create 1 to 1 Foreign Key Relationship
ALTER TABLE pbs_Shift
	ADD CONSTRAINT FK_Lead FOREIGN KEY(LeadOperatorID) REFERENCES
pbs_Employee(EmployeeID)
GO

-- Insert Production Employees
INSERT INTO pbs_Employee (FirstName, LastName, ShiftID, isForkliftCertified)
VALUES ('Scott','Snow',2,1),
('Jordan','Mangus',2,1),
('Kirk','Cruz',2,1),
('Michael','Winegarden',2,0),
('Michael','Hammock',2,0),
('Daniel','Wiley',2,0),
('Edward','Kent',2,0),
('Pon','Louen',2,1),
('Charlie','Rodriguez',1,0),
('Corey','Williams',1,0),
('Craig','Fisher',1,0),
```

```
('Mack','Varns',1,0),
('Nuth','Him',3,1),
('Pat','Cooper',3,0),
('Joeseph','Kitchichchang',3,0),
('Andrew','Quickeden',1,0),
('Brek','Hanson',1,1),
('Steve','Stewart',1,1),
('Michael','Lockhart',1,0),
('Scott','Farmer',1,0),
('Kris','Russell',3,1),
('Steven','Degenstein',3,1),
('Ron','Logan',3,1)

-- Create Certified Employee relationships

INSERT INTO pbs_AreaCertification (pbs_AreaCertification.EmployeeID,
pbs_AreaCertification.AreaID)
VALUES (21, 1), (2, 1), (1, 3), (1, 1), (5, 1), (26, 3), (26, 5), (25, 3), (25, 5), (18,
3), (12, 3), (22, 5), (17, 3)

-- Sales and Project Managers
INSERT INTO pbs_Employee (FirstName, LastName, ShiftID, isForkliftCertified)
VALUES ('Rod', 'Hatton', 4, 0),
       ('Phil', 'Ligon', 4, 0),
       ('Bim', 'Fischer', 4, 0),
       ('Todd', 'Bell', 4, 0),
       ('Matt', 'Karnes', 4, 0),
       ('Mike', 'Karnes', 4, 0),
       ('Drew', 'Cummings', 4, 0),
       ('Ryan', 'Stowers', 4, 0),
       ('John', 'Vanderhoof', 4, 0)

-- Initial Job Set
INSERT INTO pbs_Job (JobName, MasterPanelQty, PanelQty, ProjectManagerID, StatusID,
ProjectNumber)
VALUES ('LAB GeoPhaze SIPS', 99, 99, 2, 2, '00000000-0000'),
('Polestar Farm Bunkhouse', 13, 34, 32, 2, '20180219-0003'),
('Plaster Cabin', 24, 61, 28, 4, '20170227-006'),
('Shaktoolik HC', 75, 140, 32, 3, '20180206-0009'),
('Shaktoolik Utilidor', 28, 56, 32, 5, '20180328-0004'),
('Salgade Shell', 16, 31, 28, 5, '20180419-0003'),
('PO-NOEL18 Artisan', 17, 40, 32, 5, '20180514-0004'),
('18'' Yurt', 3, 5, 32, 5, '20180411-003'),
('Seibold Hangar', 19, 44, 32, 2, '20180126-004'),
('Christianson Residence', 47, 174, 32, 2, '201604140012')

--Initial Production Records 5/23-5/26
INSERT INTO pbs_AreaProductionRecord (AreaID, ProductionDate, ProductionHours, Panels,
PressesIO, ShiftID)
VALUES (4, '5/25/2018', 9.75, 34, 15, 2),
(3, '5/25/2018', 8, 30, 11, 2),
(1, '5/25/2018', 10.75, 0, 28, 2),
(1, '5/24/2018', 10.75, 0, 35, 2),
(1, '5/23/2018', 10.75, 0, 23, 2),
(4, '5/24/2018', 9.5, 31, 19, 2),
(3, '5/24/2018', 10.75, 32, 16, 2),
(1, '5/26/2018', 10.75, 0, 17, 1),
(4, '5/26/2018', 10.75, 25, 16, 1)
```

```
-- Initial Job Production Records
INSERT INTO pbs_JobAreaProductionRecord (JobID, RecordID, TotalHours)
VALUES (7, 1, 1.5), (2, 1, 1.5), (4, 1, 6.75), (9, 2, 4 ),
(2, 2, 4), (4, 3, 4), (10, 3, 1.5), (2, 3, 1),
(9, 3, 1), (7, 3, 1.5), (3, 3, 1.75), (4, 4, 2.15),
(8, 4, 1.536), (1, 4, 3.071), (9, 4, 1.842), (7, 4, 2.15),
(5, 5, 3.74), (1, 5, 1.87), (4, 5, 2.8), (6, 5, 2.34),
(8, 6, 4), (7, 6, 3.5), (4, 6, 3), (1, 7, 2),
(9, 7, 8.75), (10, 8, 4.43), (4, 8, 6.32), (4, 9, 10.75)


-- Initial Employee Station Lists


INSERT INTO pbs_AreaProductionRecordEmployeeList (RecordID, EmployeeID, StationID)
VALUES (9, 15, 26), (9, 16, 27), (9, 13, 28), (9, 24, 30), (9, 14, 31), (9, 2, 23),
(8, 23, 4), (8, 20, 6), (8, 21, 9), (8, 22, 2), (8, 12, 3),
(7, 10, 15), (7, 9, 13), (7, 7, 17), (7, 27, 23),
(6, 8, 26), (6, 25, 27), (6, 18, 28),
(5, 5, 4), (5, 14, 6), (5, 7, 9), (5, 12, 2), (5, 11, 3),
(4, 5, 4), (4, 14, 6), (4, 6, 9), (4, 12, 2), (4, 11, 3),
(3, 5, 4), (3, 24, 6), (3, 6, 9), (3, 12, 2), (3, 11, 3),
(2, 10, 15), (2, 18, 17), (2, 27, 23), (2, 9, 13),
(1, 8, 26), (1, 7, 28), (1, 26, 30)
```

Views, Functions, Procedures

This query includes the creation of base level functions, procedures and views including Get functions for ID's, Simple shift and employee views, and simple add procedures for non-constant tables. Additionally, some EXEC examples are at the bottom.

```
IF OBJECT_ID('dbo.getShiftID') IS NOT NULL
   DROP FUNCTION dbo.getShiftID
GO
IF OBJECT_ID('dbo.getStatusID') IS NOT NULL
   DROP FUNCTION dbo.getStatusID
GO
IF OBJECT_ID('dbo.getAreaID') IS NOT NULL
   DROP FUNCTION dbo.getAreaID
GO
IF OBJECT_ID('dbo.getEmpID') IS NOT NULL
   DROP FUNCTION dbo.getEmpID
GO
IF OBJECT_ID('dbo.getJobID') IS NOT NULL
   DROP FUNCTION dbo.getJobID
GO
IF OBJECT_ID('dbo.getStationID') IS NOT NULL
   DROP FUNCTION dbo.getStationID
GO
IF OBJECT_ID('dbo.getRecID') IS NOT NULL
   DROP FUNCTION dbo.getRecID
```

```
GO

-- Get ID functions


CREATE FUNCTION dbo.getShiftID(@shiftname char(10))
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_Shift.ShiftID FROM pbs_Shift
	WHERE pbs_Shift.ShiftName = @shiftname

	RETURN @ret
END
GO

CREATE FUNCTION dbo.getStatusID(@statusname varchar(15))
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_JobStatus.StatusID FROM pbs_JobStatus
	WHERE pbs_JobStatus.StatusText = @statusname

	RETURN @ret
END
GO

ALTER TABLE pbs_Job ADD DEFAULT dbo.getStatusID('Pre-Production') FOR StatusID
GO

CREATE FUNCTION dbo.getAreaID(@areaname char(15))
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_Area.AreaID FROM pbs_Area
	WHERE pbs_Area.AreaName = @areaname

	RETURN @ret
END
GO

CREATE FUNCTION dbo.getEmpID(@lastname varchar(25), @firstname varchar(25))
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_Employee.EmployeeID FROM pbs_Employee
	WHERE pbs_Employee.LastName = @lastname AND pbs_Employee.FirstName = @firstname

	RETURN @ret
END
GO

CREATE FUNCTION dbo.getStationID(@stationname varchar(30), @areaID int)
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_Station.StationID FROM pbs_Station
	WHERE pbs_Station.StationName = @stationname AND pbs_Station.AreaID = @areaID
```

```
	RETURN @ret
END
GO


CREATE FUNCTION dbo.getRecID(@date date, @shiftID int, @areaID int)
RETURNS INT AS
BEGIN
	DECLARE @ret int

	SELECT @ret = pbs_AreaProductionRecord.RecordID FROM pbs_AreaProductionRecord
	WHERE pbs_AreaProductionRecord.ProductionDate = @date
	AND pbs_AreaProductionRecord.ShiftID = @shiftID
	AND pbs_AreaProductionRecord.AreaID = @areaID

	RETURN @ret
END
GO

CREATE FUNCTION dbo.getJobID(@projectNo varchar(40))
RETURNS INT AS
BEGIN
	DECLARE @ret int
	SELECT @ret = pbs_Job.JobID FROM pbs_Job
	WHERE pbs_Job.ProjectNumber = @projectNo
	RETURN @ret
END
GO

DECLARE @this decimal(5,2) = dbo.TotalJobHours('20180328-0004')
SELECT @this AS This
IF OBJECT_ID('pbs_addNewEmployee') IS NOT NULL
   DROP PROCEDURE pbs_addNewEmployee
GO
IF OBJECT_ID('pbs_certifyEmployee') IS NOT NULL
   DROP PROCEDURE pbs_certifyEmployee
GO
IF OBJECT_ID('pbs_addNewForkLiftCert') IS NOT NULL
   DROP PROCEDURE pbs_addNewForkLiftCert
GO
IF OBJECT_ID('pbs_addNewJob') IS NOT NULL
   DROP PROCEDURE pbs_addNewJob
GO
IF OBJECT_ID('pbs_CompleteJob') IS NOT NULL
   DROP PROCEDURE pbs_CompleteJob
GO
IF OBJECT_ID('pbs_addNewRecord') IS NOT NULL
   DROP PROCEDURE pbs_addNewRecord
GO
IF OBJECT_ID('pbs_addNewJobRecord') IS NOT NULL
   DROP PROCEDURE pbs_addNewJobRecord
GO
IF OBJECT_ID('pbs_addNewRecordEmployee') IS NOT NULL
   DROP PROCEDURE pbs_addNewRecordEmployee
GO
--Add Procedures
-- New Employee
```

```
CREATE PROCEDURE pbs_addNewEmployee(@firstname varchar(25), @lastname varchar(25), @shift
char(10))
AS
BEGIN
	DECLARE @shiftid int = dbo.getShiftID(@shift)
	INSERT INTO pbs_Employee (FirstName, LastName, ShiftID, isForkliftCertified)
	VALUES (@firstname, @lastname, @shiftid, 0)
END
GO
-- Certifies employee
CREATE PROCEDURE pbs_certifyEmployee(@date date, @area char(15), @lastname varchar(25),
@firstname varchar(25))
AS
BEGIN
	DECLARE @empID int = dbo.getEmpID(@lastname, @firstname)
	DECLARE @areaID int = dbo.getAreaID(@area)
	INSERT INTO pbs_AreaCertification (EmployeeID, AreaID, DateCertified)
	VALUES (@empID, @areaID, @date)
END
GO
-- Changes Forklift Certification
CREATE PROCEDURE pbs_addNewForkLiftCert(@lastname varchar(25), @firstname varchar(25))
AS
BEGIN
	DECLARE @empID int = dbo.getEmpID(@lastname, @firstname)
	UPDATE pbs_Employee SET isForkliftCertified = 1 WHERE EmployeeID = @empID
END
GO
-- New Job with default estimates
CREATE PROCEDURE pbs_addNewJob(@projectNo varchar(40), @pmfirst varchar(25), @pmlast
varchar(25), @panelQty int, @masterpQty int, @jobname varchar(50))
AS
BEGIN
	INSERT INTO pbs_Job (JobName, MasterPanelQty, PanelQty, ProjectManagerID,
ProjectNumber)
	VALUES (@jobname, @masterpQty, @panelQty, dbo.getEmpID(@pmfirst, @pmlast),
@projectNo)
END
GO
-- Add completed values to a job
CREATE PROCEDURE pbs_CompleteJob(@projectNo varchar(40), @fmc decimal(11,2), @fr
decimal(11,2), @fo decimal(11,2))
AS
BEGIN
	UPDATE pbs_Job SET FinalMaterialCost = @fmc, FinalRevenue = @fr, FinalOverhead =
@fo, StatusID = dbo.getStatusID('Completed')
	WHERE pbs_Job.ProjectNumber = @projectNo
END
GO
-- The bulkiest section, Adds a new record
CREATE PROCEDURE pbs_addNewRecord(@date date, @shift char(10), @area char(15),
@totalhours decimal(4,2), @panels int, @pressesIO int)
AS
BEGIN
	DECLARE @shiftID int = dbo.getShiftID(@shift)
	DECLARE @areaID int = dbo.getAreaID(@area)
	INSERT INTO pbs_AreaProductionRecord (AreaID, ProductionDate, ProductionHours,
Panels, PressesIO, ShiftID)
```

```
	VALUES (@areaID, @date, @totalhours, @panels, @pressesIO, @shiftID)
END
GO
-- Adds a Job Record instance
CREATE PROCEDURE pbs_addNewJobRecord(@shift char(10), @area char(15), @date date, @jobPN
varchar(40), @jobhours decimal(4,2), @status varchar(15))
AS
BEGIN
	DECLARE @shiftID int = dbo.getShiftID(@shift)
	DECLARE @areaID int = dbo.getAreaID(@area)
	DECLARE @recID int = dbo.getRecID(@date, @shiftID, @areaID)
	DECLARE @statusID int = dbo.getStatusID(@status)
	DECLARE @jobID int = dbo.getJobID(@jobPN)
	INSERT INTO pbs_JobAreaProductionRecord(JobID, RecordID, TotalHours)
	VALUES (@jobID, @recID, @jobhours)
	UPDATE pbs_Job SET pbs_Job.StatusID = @statusID, pbs_Job.StatusChangeDate = @date
WHERE pbs_Job.JobID = @jobID
END
GO
-- Adds an Employee Record instance.
CREATE PROCEDURE pbs_addNewRecordEmployee(@shift char(10), @area char(15), @date date,
@station varchar(30), @empfn varchar(25), @empln varchar(25))
AS
BEGIN
	DECLARE @shiftID int = dbo.getShiftID(@shift)
	DECLARE @areaID int = dbo.getAreaID(@area)
	DECLARE @recID int = dbo.getRecID(@date, @shiftID, @areaID)
	DECLARE @stationID int = dbo.getStationID(@station, @areaID)
	DECLARE @empID int = dbo.getEmpID(@empfn, @empln)
	INSERT INTO pbs_AreaProductionRecordEmployeeList (RecordID, EmployeeID, StationID)
	VALUES (@recID, @empID, @stationID)
END
GO

IF OBJECT_ID('pbs_FullRoster') IS NOT NULL
   DROP VIEW pbs_FullRoster
GO
IF OBJECT_ID('pbs_ShiftLeads') IS NOT NULL
   DROP VIEW pbs_ShiftLeads
GO
IF OBJECT_ID('pbs_CertifiedOperators') IS NOT NULL
   DROP VIEW pbs_CertifiedOperators
GO
IF OBJECT_ID('pbs_ForkliftOperators') IS NOT NULL
   DROP VIEW pbs_ForkliftOperators
GO
IF OBJECT_ID('pbs_AShift') IS NOT NULL
   DROP VIEW pbs_AShift
GO
IF OBJECT_ID('pbs_BShift') IS NOT NULL
   DROP VIEW pbs_BShift
GO
IF OBJECT_ID('pbs_CShift') IS NOT NULL
   DROP VIEW pbs_CShift
GO
IF OBJECT_ID('pbs_CompletedJobs') IS NOT NULL
   DROP VIEW pbs_CompletedJobs
GO
```

```
IF OBJECT_ID('pbs_InProduction') IS NOT NULL
   DROP VIEW pbs_InProduction
GO
IF OBJECT_ID('pbs_PreProduction') IS NOT NULL
   DROP VIEW pbs_PreProduction
GO
IF OBJECT_ID('pbs_OnHold') IS NOT NULL
   DROP VIEW pbs_OnHold
GO
-- Common Views
-- See Employee with shift listing.
GO
CREATE VIEW pbs_FullRoster AS
SELECT LastName, FirstName, ShiftName AS [Shift],
 ROW_NUMBER() OVER (ORDER BY LastName ASC) AS RowNum
FROM pbs_Employee
JOIN pbs_Shift ON pbs_Employee.ShiftID = pbs_Shift.ShiftID
GO
-- See Shift Leads
CREATE VIEW pbs_ShiftLeads AS
SELECT pbs_Shift.ShiftName
, pbs_Employee.FirstName + ' ' + pbs_Employee.LastName AS Lead
FROM pbs_Shift
JOIN pbs_Employee ON pbs_Shift.LeadOperatorID = pbs_Employee.EmployeeID
GO
-- See Certified Operators
CREATE VIEW pbs_CertifiedOperators AS
SELECT pbs_Employee.LastName + ', ' + pbs_Employee.FirstName AS Operator,
pbs_Area.AreaName AS Area,
ROW_NUMBER() OVER (ORDER BY pbs_Area.AreaName ASC) AS RowNum
FROM pbs_AreaCertification
JOIN pbs_Area ON pbs_Area.AreaID = pbs_AreaCertification.AreaID
JOIN pbs_Employee ON pbs_Employee.EmployeeID = pbs_AreaCertification.EmployeeID
GO
-- See forklift operators
CREATE VIEW pbs_ForkliftOperators AS
SELECT pbs_Employee.LastName + ', ' + pbs_Employee.FirstName AS [Forklift Operators],
pbs_Shift.ShiftName,
ROW_NUMBER() OVER (ORDER BY pbs_Shift.ShiftName ASC) AS RowNum
FROM pbs_Employee
JOIN pbs_Shift ON pbs_Employee.ShiftID = pbs_Shift.ShiftID
WHERE isForkliftCertified = 1
GO
-- A Shift
CREATE VIEW pbs_AShift AS
SELECT pbs_Employee.FirstName + ' ' + pbs_Employee.LastName AS [Name]
FROM pbs_Employee
WHERE ShiftID = 1
GO
-- B Shift
CREATE VIEW pbs_BShift AS
SELECT pbs_Employee.FirstName + ' ' + pbs_Employee.LastName AS [Name]
FROM pbs_Employee
WHERE ShiftID = 2
GO
-- C Shift
CREATE VIEW pbs_CShift AS
SELECT pbs_Employee.FirstName + ' ' + pbs_Employee.LastName AS [Name]
```

```
FROM pbs_Employee
WHERE ShiftID = 3
GO
-- Completed Jobs
CREATE VIEW pbs_CompletedJobs AS
SELECT pbs_Job.JobName, pbs_Job.ProjectNumber, pbs_Job.FinalRevenue,
pbs_Job.StatusChangeDate
FROM pbs_Job
WHERE pbs_Job.StatusID = dbo.getStatusID('Completed')
GO
-- In-Production Jobs
CREATE VIEW pbs_InProduction AS
SELECT pbs_Job.JobName, pbs_Job.ProjectNumber, pbs_Job.EstRevenue,
pbs_Job.StatusChangeDate
FROM pbs_Job
WHERE pbs_Job.StatusID = dbo.getStatusID('In Progress') OR pbs_Job.StatusID =
dbo.getStatusID('Started')
GO
-- Pre-Production Jobs
CREATE VIEW pbs_PreProduction AS
SELECT pbs_Job.JobName, pbs_Job.ProjectNumber, pbs_Job.EstRevenue,
pbs_Job.StatusChangeDate
FROM pbs_Job
WHERE pbs_Job.StatusID = dbo.getStatusID('Pre-Production')
GO
-- On Hold Jobs
CREATE VIEW pbs_OnHold AS
SELECT pbs_Job.JobName, pbs_Job.ProjectNumber, pbs_Job.EstRevenue,
pbs_Job.StatusChangeDate
FROM pbs_Job
WHERE pbs_Job.StatusID = dbo.getStatusID('On Hold')
GO


-- EXEC Statements
EXEC pbs_addNewEmployee 'Justin', 'Kubeck', 'B Shift'
EXEC pbs_addNewEmployee 'Curtis', 'Slater', 'Office'
```

Aggregate Functions

These functions are primarily for answering the initial questions for the database. The first group builds up to displaying a time and dollar values summary for each job in the database along with a row of correlation values at the bottom. The second group adds up all the hours worked at each station. The primary goal for this view is to be exported and used to create a regression model.

```
IF OBJECT_ID('dbo.TotalJobHours') IS NOT NULL
   DROP FUNCTION dbo.TotalJobHours
GO
-- Calculates how long it took to complete a job
CREATE FUNCTION dbo.TotalJobHours(@projectNo varchar(50))
RETURNS DECIMAL(5,2) AS
BEGIN
```

```
	DECLARE @ret Decimal(5,2)
	DECLARE @jobID int = dbo.getJobID(@projectNo)
	SELECT @ret = SUM(pbs_JobAreaProductionRecord.TotalHours)
	FROM pbs_JobAreaProductionRecord WHERE JobID = @jobID
	RETURN @ret
END
GO


IF OBJECT_ID('pbs_JobHoursSummary') IS NOT NULL
   DROP VIEW pbs_JobHoursSummary
GO
-- shows a summary of each job and how long that job took along with varioud final dollar
figures
CREATE VIEW pbs_JobHoursSummary AS
SELECT pbs_Job.JobName
	, pbs_Job.ProjectNumber
	, pbs_Job.FinalRevenue
	, dbo.TotalJobHours(pbs_Job.ProjectNumber) AS TotalHours
	, pbs_Job.FinalMaterialCost
	, pbs_Job.FinalOverhead
	FROM pbs_Job
	WHERE pbs_Job.StatusID = dbo.getStatusID('Completed')
GO

IF OBJECT_ID('pbs_JobCorrelationSummary') IS NOT NULL
   DROP VIEW pbs_JobCorrelationSummary
GO
-- Formula for Pearson correlation retrieved from
https://www.mssqltips.com/sqlservertip/4544/calculating-the-pearson-product-moment-
correlation-coefficient-in-tsql/ on 6/28/18
-- shows correlation coefficients for Final Revenue, and each of the three identified
contributing factors: labor, materials, overhead
CREATE VIEW pbs_JobCorrelationSummary AS
SELECT
	'Correlation Summary' AS Summary
	, ' ' AS ' '
	, SUM(FinalRevenue) AS Total
	, (Avg(FinalRevenue * dbo.TotalJobHours(pbs_Job.ProjectNumber)) -
(Avg(FinalRevenue) * Avg(dbo.TotalJobHours(pbs_Job.ProjectNumber)))) /
(StDevP(FinalRevenue) * StDevP(dbo.TotalJobHours(pbs_Job.ProjectNumber))) AS
RevenueAndTotalHours
	, (Avg(FinalRevenue * FinalMaterialCost) - (Avg(FinalRevenue) *
Avg(FinalMaterialCost)))/ (StDevP(FinalRevenue) * StDevP(FinalMaterialCost)) AS
RevenueAndMaterials
	, (Avg(FinalRevenue * FinalOverhead) - (Avg(FinalRevenue) * Avg(FinalOverhead)))/
(StDevP(FinalRevenue) * StDevP(FinalOverhead)) AS RevenueAndOverhead
FROM pbs_Job WHERE pbs_Job.StatusID = dbo.getStatusID('Completed')
GO

IF OBJECT_ID('pbs_viewCompletedJobSummary') IS NOT NULL
   DROP VIEW pbs_viewCompletedJobSummary
GO
-- Combines the previous two views to be viewed as one
CREATE VIEW pbs_viewCompletedJobSummary AS
SELECT * FROM pbs_JobHoursSummary
UNION ALL
SELECT * FROM pbs_JobCorrelationSummary
```

```
GO
```

Output for pbs_viewCompletedJobSummary

Results | Messages

	JobName	ProjectNumber	FinalRevenue	TotalHours	FinalMaterialCost	FinalOverhead
1	LAB GeoPhaze SIPS	00000000-0000	5656.59	11.94	5465	215
2	Polestar Farm Bunkhouse	20180219-0003	156684.00	8.78	156	12356
3	Plaster Cabin	20170227-006	5848.00	19.85	1232	2132
4	Shaktoolik HC	20180206-0009	12.00	48.61	123	9858
5	Shaktoolik Utilidor	20180328-0004	898.00	7.74	77	8595
6	Salgade Shell	20180419-0003	1561.02	8.09	1848.56	26
7	PO-NOEL18 Artisan	20180514-0004	156.00	8.65	3165	9981.65
8	18' Yurt	20180411-003	2315.00	5.54	1516	418165.15
9	Seibold Hangar	20180126-004	894.00	15.59	84	6516
10	Christianson Residence	201604140012	878.00	65.14	189	9859
11	Phillips	20180516-0001	4324.00	3.18	23142	53463
12	Bone / Filson	20180223-0003	65765.00	4.25	2645	342
13	Carter	20170616-0003	4321.00	26.1	23435	12345
14	Correlation Summary		249312.61	-0.238361734189978	-0.170293448968514	-0.116977491977056

****Final Revenue, Final Material Cost, and Final Overhead currently contains dummy data, these are not actual values**

```
IF OBJECT_ID('dbo.StationHoursForDay') IS NOT NULL
   DROP FUNCTION dbo.StationHoursForDay
GO
-- returns the total hours among areas spent at respective stations for a specific day
CREATE FUNCTION dbo.StationHoursForDay(@date date, @station varchar(30))
RETURNS DECIMAL(5,2) AS
BEGIN
       DECLARE @ret Decimal(5,2)
       SELECT @ret = SUM(pbs_AreaProductionRecord.ProductionHours)
       FROM pbs_AreaProductionRecord
       JOIN pbs_AreaProductionRecordEmployeeList ON pbs_AreaProductionRecord.RecordID =
pbs_AreaProductionRecordEmployeeList.RecordID
       JOIN pbs_Station ON pbs_AreaProductionRecordEmployeeList.StationID =
pbs_Station.StationID
       WHERE ProductionDate = @date AND pbs_Station.StationName = @station
       RETURN @ret
END
GO

IF OBJECT_ID('dbo.ProductionSpeedForDay') IS NOT NULL
   DROP FUNCTION dbo.ProductionSpeedForDay
GO
-- returns the overall speed Panels/Hour for a given day. In real operations, the panels
are weighted based on complexity but that is not included here
CREATE FUNCTION dbo.ProductionSpeedForDay(@date date)
RETURNS DECIMAL(5,2) AS
BEGIN
       DECLARE @panels Decimal(5,2)
       DECLARE @hours Decimal(5,2)
       SELECT @hours = SUM(pbs_AreaProductionRecord.ProductionHours)
```

```
	FROM pbs_AreaProductionRecord
	WHERE ProductionDate = @date AND pbs_AreaProductionRecord.AreaID = 3 OR
pbs_AreaProductionRecord.AreaID = 4 OR pbs_AreaProductionRecord.AreaID = 5
	SELECT @panels = SUM(pbs_AreaProductionRecord.Panels)
	FROM pbs_AreaProductionRecord
	WHERE ProductionDate = @date AND pbs_AreaProductionRecord.AreaID = 3 OR
pbs_AreaProductionRecord.AreaID = 4 OR pbs_AreaProductionRecord.AreaID = 5
	RETURN @panels/@hours
END
GO
-- Shows every day in the database with the hours for each respective station as well as
Panels per hour for that day. Specifically to be exported as a CSV into Excel to run
Regression
IF OBJECT_ID('pbs_StationHoursReport') IS NOT NULL
   DROP VIEW pbs_StationHoursReport
GO
CREATE VIEW pbs_StationHoursReport
AS
SELECT pbs_AreaProductionRecord.ProductionDate
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Layout'),0) AS Layout
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Cutter'),0) AS Cutter
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Hogger'),0) AS Hogger
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Scrape and Edge Seal 1'),0) AS [S/ES 1]
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Scrape and Edge Seal 2'),0) AS [S/ES 2]
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'QC'),0) AS QC
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Lumber Assistant'),0) AS LumberAssist
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'Lumber Install'),0) AS LumberInstall
, ISNULL(dbo.StationHoursForDay(ProductionDate, 'CA Stickers'),0) AS CAStickers
, dbo.ProductionSpeedForDay(ProductionDate) AS PanelsPerHour
FROM pbs_AreaProductionRecord
	JOIN pbs_AreaProductionRecordEmployeeList ON pbs_AreaProductionRecord.RecordID =
pbs_AreaProductionRecordEmployeeList.RecordID
	JOIN pbs_Station ON pbs_AreaProductionRecordEmployeeList.StationID =
pbs_Station.StationID
	WHERE pbs_AreaProductionRecord.AreaID = 3 OR pbs_AreaProductionRecord.AreaID = 4
OR pbs_AreaProductionRecord.AreaID = 5
GROUP BY ProductionDate
GO
```

Regression Output for pbs_StationHoursReport: View output imported as CSV to Excel

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.993676656
R Square	0.987393297
Adjusted R Square	0.96217989
Standard Error	0.012605018
Observations	10

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	6	0.03733334	0.00622222	39.1614397	0.00609944
Residual	3	0.00047666	0.00015889		
Total	9	0.03781			

	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	3.873606707	0.07268358	53.2941106	1.4551E-05	3.64229512	4.10491829
Layout	-0.012286849	0.00179862	-6.8312814	0.00641854	-0.0180108	-0.0065629
Cutter	0.003103094	0.00101574	3.05501598	0.0552093	-0.0001294	0.00633562
Hogger	-0.018864504	0.00205023	-9.201165	0.00271504	-0.0253893	-0.0123398
S/ES 1	-0.016757261	0.00139971	-11.971963	0.00125364	-0.0212118	-0.0123028
QC	-0.006396278	0.00521672	-1.2261119	0.30762171	-0.0229982	0.01020564
Lumber Install	-0.018492436	0.00202541	-9.130238	0.00277703	-0.0249382	-0.0120467

Users and Permissions

The three users created for this database are for the production leaders, project managers, and upper management.

```
-- Login and User Creation. Generated Scripts from SSMS
USE [master]
GO
CREATE LOGIN [pbsProjects] WITH PASSWORD=N'123456' MUST_CHANGE,
DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
USE [PBS]
GO
CREATE USER [pbsProjects] FOR LOGIN [pbsProjects]
GO

USE [master]
GO
CREATE LOGIN [pbsProduction] WITH PASSWORD=N'123456' MUST_CHANGE,
DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
USE [PBS]
```

```
GO
CREATE USER [pbsProduction] FOR LOGIN [pbsProduction]
GO

USE [master]
GO
CREATE LOGIN [pbsManagement] WITH PASSWORD=N'123456' MUST_CHANGE,
DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
USE [PBS]
GO
CREATE USER [pbsManagement] FOR LOGIN [pbsManagement]
GO

-- GRANTS for each User

-- Project Management can view Job related reports as well as add new jobs and enter
closing information for post production jobs.
GRANT SELECT ON pbs_CompletedJobs TO pbsManagement
GRANT SELECT ON pbs_InProduction TO pbsManagement
GRANT SELECT ON pbs_PreProduction  TO pbsManagement
GRANT SELECT ON pbs_OnHold         TO pbsManagement
GRANT SELECT ON pbs_JobHoursSummary       TO pbsProject
GRANT EXECUTE ON TotalJobHours     TO pbsProject
GRANT EXECUTE ON pbs_addNewJob     TO pbsProject
GRANT EXECUTE ON pbs_CompleteJob TO pbsProject

-- Production leadership can view job reports related to production, can execute job-
production related functions and can execute functions to add new records.
GRANT SELECT ON pbs_FullRoster     TO pbsProduction
GRANT SELECT ON pbs_ShiftLeads     TO pbsProduction
GRANT SELECT ON pbs_CertifiedOperators          TO pbsProduction
GRANT SELECT ON pbs_ForkliftOperators           TO pbsProduction
GRANT SELECT ON pbs_AShift         TO pbsProduction
GRANT SELECT ON pbs_BShift         TO pbsProduction
GRANT SELECT ON pbs_CShift         TO pbsProduction
GRANT SELECT ON pbs_CompletedJobs TO pbsProduction
GRANT SELECT ON pbs_InProduction TO pbsProduction
GRANT SELECT ON pbs_PreProduction  TO pbsProduction
GRANT SELECT ON pbs_OnHold         TO pbsProduction
GRANT SELECT ON pbs_StationHoursReport    TO pbsProduction
GRANT SELECT ON pbs_JobHoursSummary       TO pbsProduction
GRANT EXECUTE ON TotalJobHours     TO pbsProduction
GRANT EXECUTE ON StationHoursForDay       TO pbsProduction
GRANT EXECUTE ON ProductionSpeedForDay  TO pbsProduction
GRANT EXECUTE ON pbs_addNewRecord  TO pbsProduction
GRANT EXECUTE ON pbs_addNewJobRecord      TO pbsProduction
GRANT EXECUTE ON pbs_addNewRecordEmployee TO pbsProduction


-- upper level management has access to all views, external functions and procedures
GRANT SELECT ON pbs_FullRoster     TO pbsManagement
GRANT SELECT ON pbs_ShiftLeads     TO pbsManagement
GRANT SELECT ON pbs_CertifiedOperators          TO pbsManagement
GRANT SELECT ON pbs_ForkliftOperators           TO pbsManagement
GRANT SELECT ON pbs_AShift         TO pbsManagement
GRANT SELECT ON pbs_BShift         TO pbsManagement
GRANT SELECT ON pbs_CShift         TO pbsManagement
```

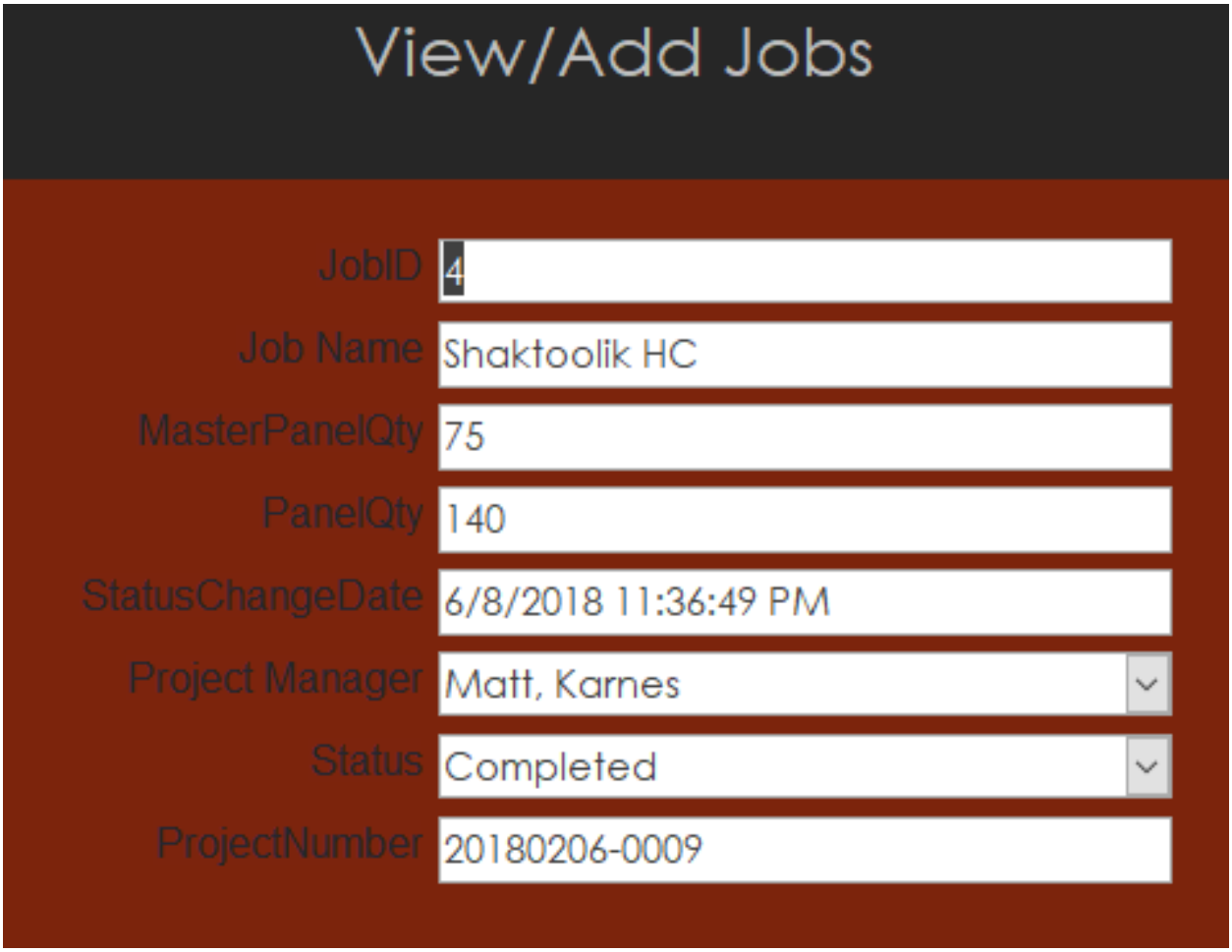
```
GRANT SELECT ON pbs_CompletedJobs TO pbsManagement
GRANT SELECT ON pbs_InProduction TO pbsManagement
GRANT SELECT ON pbs_PreProduction  TO pbsManagement
GRANT SELECT ON pbs_OnHold         TO pbsManagement
GRANT SELECT ON pbs_StationHoursReport    TO pbsManagement
GRANT SELECT ON pbs_JobHoursSummary       TO pbsManagement
GRANT SELECT ON pbs_JobCorrelationSummary TO pbsManagement
GRANT SELECT ON pbs_viewCompletedJobSummary     TO pbsManagement
GRANT EXECUTE ON TotalJobHours     TO pbsManagement
GRANT EXECUTE ON StationHoursForDay       TO pbsManagement
GRANT EXECUTE ON ProductionSpeedForDay  TO pbsManagement
GRANT EXECUTE ON pbs_CompleteJob   TO pbsManagement
GRANT EXECUTE ON pbs_addNewRecord  TO pbsManagement
GRANT EXECUTE ON pbs_addNewJobRecord      TO pbsManagement
GRANT EXECUTE ON pbs_addNewRecordEmployee TO pbsManagement
GRANT EXECUTE ON pbs_addNewEmployee       TO pbsManagement
GRANT EXECUTE ON pbs_certifyEmployee      TO pbsManagement
GRANT EXECUTE ON pbs_addNewForkLiftCert   TO pbsManagement
GRANT EXECUTE ON pbs_addNewJob     TO pbsManagement

-- Currently no procedure for removing an employee who was terminated or left
GRANT EXECUTE ON getEmpID TO pbsManagement
GRANT DELETE ON pbs_Employee to pbsManagement


-- veiws all objects to assign user levels.
SELECT  *
FROM sys.objects
WHERE   type = 'V' OR type = 'FN' OR type = 'P' OR type = 'AF'
```

Forms and Reports

Two forms and one report were created for the purposes of interfacing with the database at a non-query level. The two forms simulate the most likely to be used procedures: adding a new Job and adding a new production record. The report answers the fourth question regarding the acquisition of a small batch laminator.


View/Add Jobs
JobID
4
Job Name
Shaktoolik HC
MasterPanelQty
75
PanelQty
140
StatusChangeDate
6/8/2018 11:36:49 PM
Project Manager
Matt, Karnes
Status
Completed
ProjectNumber
20180206-0009

New Batch Laminator Report

ProductionDate	ProductionHours		Panels	PressesIO		
2018-05-25		10.75	0	28	168	451.5
2018-05-24		10.75	0	35	210	451.5
2018-05-23		10.75	0	23	138	451.5
2018-05-26		10.75	0	17	102	451.5
2018-05-27		10.75	0	14	84	451.5
2018-06-02		3	0	6	36	126
2018-06-01		9.75	0	26	156	409.5
2018-05-29		10.75	0	27	162	451.5
2018-05-30		10.75	0	28	168	451.5

Conclusion

There is no benefit to having a seperate small batch laminator

Monday, July 2, 2018 Page 1 of 1