|  |
| --- |
|  |
| Revenue Improvement Database |
| Database project for 0406 - IST659 |
|  |
| **Scott Snow** |
| **6/29/2018** |

**Table of Contents**

|  |  |
| --- | --- |
| 1. **Summary** | Page 2 |
| 1. **Conceptual Model** | Page 4 |
| 1. **Logical Model** | Page 6 |
| 1. **Table Creation Data** | Page 8 |
| 1. **Constant Data Insert** | Page 11 |
| 1. **Initial Data Insert** | Page 12 |
| 1. **Views, Procedures, Functions** | Page 14 |
| 1. **Aggregate Functions** | Page 20 |
| 1. **Logins, Users and Permissions** | Page 24 |
| 1. **Forms and Reports** | Page 27 |

**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.

1. 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.

1. 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.

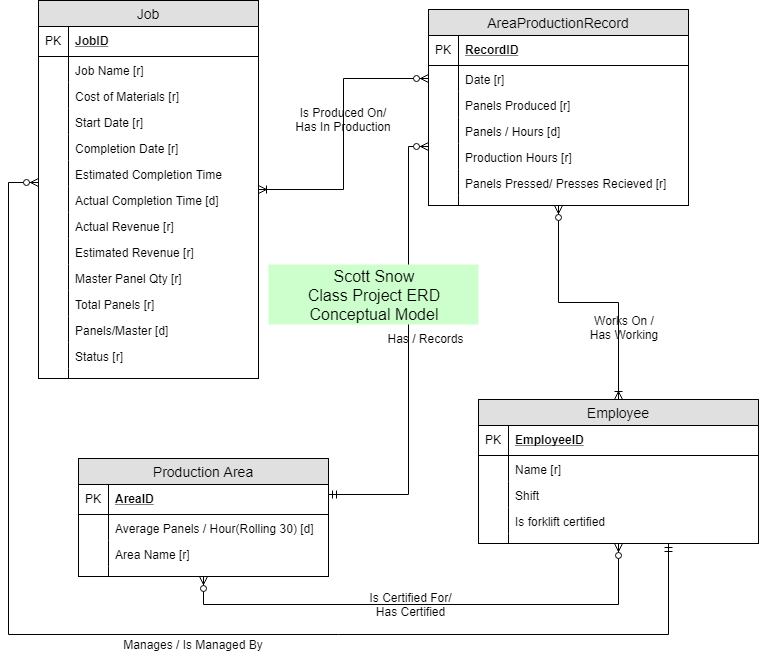
1. Is the time spent doing *non-price increasing but still customer added activities* at a reasonable amount.

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

1. 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.

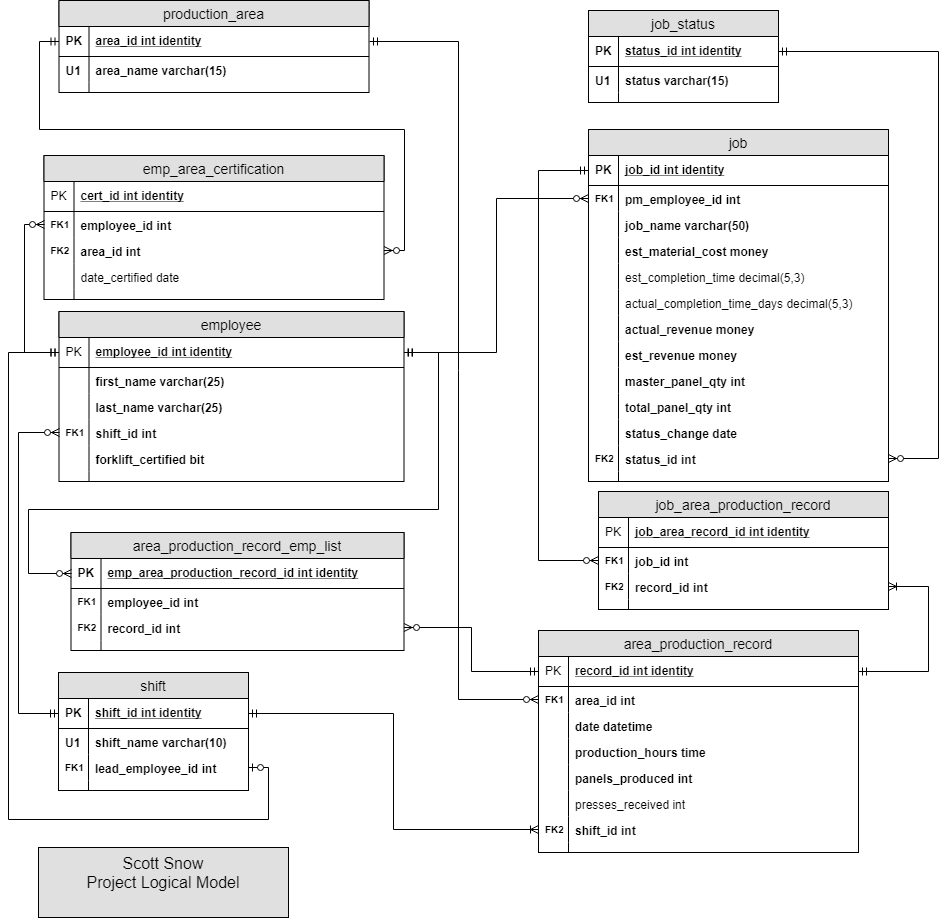
Each job is managed by a project manager. Each job has a production status. The four production statuses are pre-production, in progress, post-production, completed. When a job is in progress, it will appear on each new production area records until it enters post-production or is put on hold. Each job 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

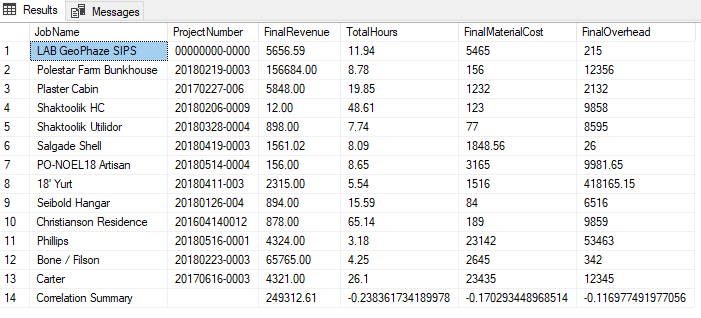
SELECT \* FROM pbs\_JobHoursSummary

UNION ALL

SELECT \* FROM pbs\_JobCorrelationSummary

GO

**Output for pbs\_viewCompletedJobSummary**



\***\*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

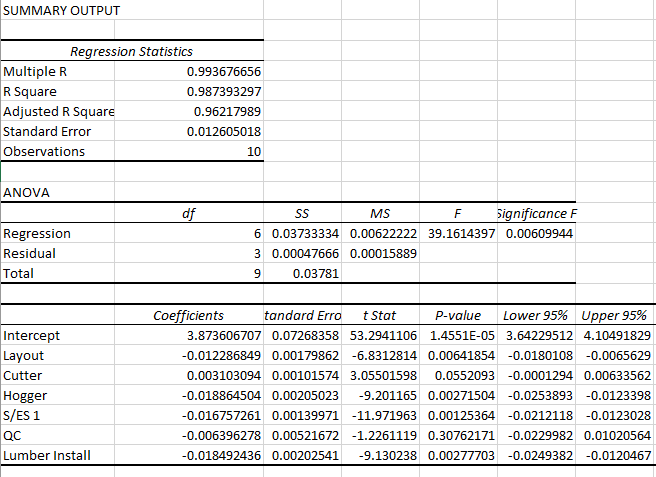
JOIN pbs\_Station ON pbs\_AreaProductionRecordEmployeeList.StationID = pbs\_Station.StationID

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**



**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.



