Create summary information about the count of timesheets per workorderID and taskID to help validate imported data

SELECT WorkOrderID,

pjTimeSheets.TaskID,

TaskDescription,

Count(\*) CountOfTimeSheets,

Sum(HoursWorked) TotalHoursWorked

FROM pjTimeSheets

left outer join pjTask

on pjTimeSheets.TaskID = pjTask.TaskID

GROUP BY WorkOrderID, pjTimeSheets.TaskID, TaskDescription

Compare estimated time and material costs to actual time and material costs for a specific order

CREATE VIEW vActualLabor AS

SELECT WorkOrderID,

TaskID,

sum(HoursWorked) ActualHours,

sum(HoursWorked\*HourlyPayRate) ActualLaborCost

FROM pjTimeSheets

GROUP BY WorkOrderID, TaskID

CREATE VIEW vActualMaterial AS

SELECT WorkOrderID,

TaskID,

FORMAT(sum(Quantity\*CostPerUOM), 'N2') ActualMaterialCost

FROM pjMaterialAssigned

GROUP By WorkOrderID, TaskID

SELECT pjWorkOrderTask.WorkOrderID,

pjWorkOrderTask.TaskID,

TaskDescription,

EstHours,

ISNULL(ActualHours,0.00) ActualHours,

EstLaborCost,

ISNULL(ActualLaborCost,0.00) ActualLaborCost,

EstMaterialCost,

ISNULL(ActualMaterialCost,0.00) ActualMaterialCost,

CASE

WHEN DateCompleted is NULL

THEN (EstHours - ISNULL(ActualHours, 0))

WHEN DateCompleted is not NULL

THEN '0.00'

END LaborToFinish,

CASE

WHEN DateCompleted is NULL

THEN (EstMaterialCost - ISNULL(ActualMaterialCost, 0))

WHEN DateCompleted is not NULL

THEN '0.00'

END MaterialToFinish,

CASE

WHEN DateStarted is Null

THEN 'Not Started'

WHEN DateStarted is not NULL

And DateCompleted is NULL

THEN 'In Process'

WHEN DateCompleted is not NULL

THEN 'Completed'

END StatusMessage

FROM pjWorkOrderTask

left outer join pjTask

on pjWorkOrderTask.TaskID = pjTask.TaskID

left outer join vActualLabor

on pjWorkOrderTask.WorkOrderID = vActualLabor.WorkOrderID

and pjWorkOrderTask.TaskID = vActualLabor.TaskID

left outer join vActualMaterial

on pjWorkOrderTask.WorkOrderID = vActualMaterial.WorkOrderID

and pjWorkOrderTask.TaskID = vActualMaterial.TaskID

WHERE pjWorkOrderTask.WorkOrderID = '16885'

**Display detailed information about all completed work orders**

SELECT pjWorkOrderTask.WorkOrderID,

pjWorkOrderTask.TaskID,

TaskDescription,

DateCompleted,

EstHours,

ISNULL(ActualHours,0.00) ActualHours,

EstLaborCost,

ISNULL(ActualLaborCost,0.00) ActualLaborCost,

EstLaborCost - ISNULL(ActualLaborCost, 0) LaborCostDiff,

EstMaterialCost,

ISNULL(ActualMaterialCost,0.00) ActualMaterialCost,

EstMaterialCost - ISNULL(ActualMaterialCost,0) MatCostDiff

FROM pjWorkOrderTask

left outer join pjTask

on pjWorkOrderTask.TaskID = pjTask.TaskID

left outer join vActualLabor

on pjWorkOrderTask.WorkOrderID = vActualLabor.WorkOrderID

and pjWorkOrderTask.TaskID = vActualLabor.TaskID

left outer join vActualMaterial

on pjWorkOrderTask.WorkOrderID = vActualMaterial.WorkOrderID

and pjWorkOrderTask.TaskID = vActualMaterial.TaskID

WHERE pjWorkOrderTask.WorkOrderID NOT IN

(SELECT WorkOrderID

FROM pjWorkOrderTask

WHERE datecompleted is null

Group by WorkOrderID)

A screenshot of a computer

Description automatically generated with medium confidence

**Display completed work orders**

SELECT pjWorkOrderTask.WorkOrderID,

CustomerName,

min(datestarted) DateStarted,

max(DateCompleted) DateCompleted,

sum(EstHours) EstHours,

sum(ISNULL(ActualHours,0.00)) ActualHours,

sum(EstLaborCost) EstLaborCost,

sum(ISNULL(ActualLaborCost,0.00)) ActualLaborCost,

sum(EstLaborCost - ISNULL(ActualLaborCost, 0)) LaborCostDiff,

sum(EstMaterialCost) EstMaterialCost,

sum(ISNULL(ActualMaterialCost,0)) ActualMaterialCost,

sum(EstMaterialCost - ISNULL(ActualMaterialCost,0)) MatCostDiff,

sum(EstLaborCost - ISNULL(ActualLaborCost, 0)) + sum(EstMaterialCost - ISNULL(ActualMaterialCost,0)) TotalCostDiff

FROM pjWorkOrderTask

left outer join pjTask

on pjWorkOrderTask.TaskID = pjTask.TaskID

left outer join vActualLabor

on pjWorkOrderTask.WorkOrderID = vActualLabor.WorkOrderID

and pjWorkOrderTask.TaskID = vActualLabor.TaskID

left outer join vActualMaterial

on pjWorkOrderTask.WorkOrderID = vActualMaterial.WorkOrderID

and pjWorkOrderTask.TaskID = vActualMaterial.TaskID

WHERE pjWorkOrderTask.WorkOrderID NOT IN

(SELECT WorkOrderID

FROM pjWorkOrderTask

WHERE datecompleted is null

Group by WorkOrderID)

GROUP BY pjWorkOrderTask.WorkOrderID, customername

**Display completed work order that was the most over budget in total**

WITH CTECompletedWork AS

(SELECT pjWorkOrderTask.WorkOrderID,

CustomerName,

min(datestarted) DateStarted,

max(DateCompleted) DateCompleted,

sum(EstHours) EstHours,

sum(ISNULL(ActualHours,0.00)) ActualHours,

sum(EstLaborCost) EstLaborCost,

sum(ISNULL(ActualLaborCost,0.00)) ActualLaborCost,

sum(EstLaborCost - ISNULL(ActualLaborCost, 0)) LaborCostDiff,

sum(EstMaterialCost) EstMaterialCost,

sum(ISNULL(ActualMaterialCost,0)) ActualMaterialCost,

sum(EstMaterialCost - ISNULL(ActualMaterialCost,0)) MatCostDiff,

sum(EstLaborCost - ISNULL(ActualLaborCost, 0)) + sum(EstMaterialCost - ISNULL(ActualMaterialCost,0)) TotalCostDiff

FROM pjWorkOrderTask

left outer join pjTask

on pjWorkOrderTask.TaskID = pjTask.TaskID

left outer join vActualLabor

on pjWorkOrderTask.WorkOrderID = vActualLabor.WorkOrderID

and pjWorkOrderTask.TaskID = vActualLabor.TaskID

left outer join vActualMaterial

on pjWorkOrderTask.WorkOrderID = vActualMaterial.WorkOrderID

and pjWorkOrderTask.TaskID = vActualMaterial.TaskID

WHERE pjWorkOrderTask.WorkOrderID NOT IN

(SELECT WorkOrderID

FROM pjWorkOrderTask

WHERE datecompleted is null

Group by WorkOrderID)

GROUP BY pjWorkOrderTask.WorkOrderID, customername)

SELECT \*

FROM cteCompletedWork

WHERE TotalCostDiff =

(SELECT MIN(TotalCostDiff)

FROM cteCompletedWork)



**Stored Procedure 1**

**Create View Table for WorkOrderID to compare estimated time and material costs to actual time and material costs for all order.**

CREATE VIEW vStatus AS

SELECT pjWorkOrderTask.WorkOrderID,

pjWorkOrderTask.TaskID,

TaskDescription,

EstHours,

ISNULL(ActualHours,0.00) ActualHours,

EstLaborCost,

ISNULL(ActualLaborCost,0.00) ActualLaborCost,

EstMaterialCost,

ISNULL(ActualMaterialCost,0.00) ActualMaterialCost,

CASE

WHEN DateCompleted is NULL

THEN (EstHours - ISNULL(ActualHours, 0))

WHEN DateCompleted is not NULL

THEN '0.00'

END LaborToFinish,

CASE

WHEN DateCompleted is NULL

THEN (EstMaterialCost - ISNULL(ActualMaterialCost, 0))

WHEN DateCompleted is not NULL

THEN '0.00'

END MaterialToFinish,

CASE

WHEN DateStarted is Null

THEN 'Not Started'

WHEN DateStarted is not NULL

And DateCompleted is NULL

THEN 'In Process'

WHEN DateCompleted is not NULL

THEN 'Completed'

END StatusMessage

FROM pjWorkOrderTask

left outer join pjTask

on pjWorkOrderTask.TaskID = pjTask.TaskID

left outer join vActualLabor

on pjWorkOrderTask.WorkOrderID = vActualLabor.WorkOrderID

and pjWorkOrderTask.TaskID = vActualLabor.TaskID

left outer join vActualMaterial

on pjWorkOrderTask.WorkOrderID = vActualMaterial.WorkOrderID

and pjWorkOrderTask.TaskID = vActualMaterial.TaskID

**Create Stored Procedure**

**The purpose of this stored procedure is to compare estimated time and material costs to actual time and material costs, as well as current status for a specific work order.**

CREATE PROCEDURE upWorkOrderStatus

AS

DECLARE @workorderin CHAR(5)

SET @workorderin = '16885'

IF

(SELECT count(\*)

FROM vStatus

WHERE workorderID = @workorderin) > 0

SELECT \*

FROM vStatus

WHERE workorderID = @workorderin

ELSE

PRINT 'No Work Order with WorkOrderID = ' + @workorderin PRINT 'No Work Order with WorkOrderID = ' + @workorderin

**Modify the stored procedure**

ALTER PROCEDURE upWorkOrderStatus

@workorderin CHAR(5)

AS

IF

(SELECT count(\*)

FROM vStatus

WHERE workorderID = @workorderin) > 0

SELECT \*

FROM vStatus

WHERE workorderID = @workorderin

ELSE

PRINT 'No Work Order with WorkOrderID = ' + @workorderin

**Stored Procedure 2**

**The purpose of this stored procedure is to create a table to automate the detection of work order that is over budget. From this table, we can know where the over budget comes from: labor, material or both.**

CREATE PROCEDURE upTotalOverBudget

AS

DROP TABLE proOverBudget

SELECT WorkOrderID,

CustomerName,

DateStarted,

DateCompleted,

LaborCostDiff,

MatCostDiff,

TotalCostDiff

INTO proOverBudget

FROM vCostDiff

WHERE workorderID is not null

And TotalCostDiff < 0

**Modify the stored procedure**

ALTER PROCEDURE upTotalOverBudget

AS

IF object\_id ('proOverBudget') is not null

BEGIN

DROP TABLE proOverBudget

END

SELECT WorkOrderID,

CustomerName,

DateStarted,

DateCompleted,

LaborCostDiff,

MatCostDiff,

TotalCostDiff

INTO proOverBudget

FROM vCostDiff

WHERE workorderID is not null

And TotalCostDiff < 0

Exec upTotalOverBudget

SELECT \*

FROM proOverBudget

**Function 1:**

**This function is to know the working date for a work order**

CREATE FUNCTION DateWorking

(@datestartedin date,

@datecompletedin date)

RETURNS int

BEGIN

RETURN datediff(dd, @datestartedin, @datecompletedin)

END

GO

SELECT WorkOrderID,

dbo.DateWorking(datestarted, datecompleted) DateWork

FROM vCostDiff

**Function 2:**

**This function is to format Employee Name**

CREATE FUNCTION FullEmplName

(@lastnamein char(20),

@firstnamein char(20))

RETURNS CHAR(40)

BEGIN

RETURN substring(@lastnamein,1, len(@lastnamein)) + ', ' + substring(@firstnamein, 1, len(@firstnamein))

END

GO

SELECT employeeID,

dbo.FullEmplName(lastname, firstname) FullName

FROM pjEmployee