Database Design and Implementation

HW 05

Team 08

Henriod, Terence

Santoyo, Jorge

Singh, Raja

February 26, 2015

/\*\*

\* IS675 - HW05

\* Using SQL Operations on Multiple Tables

\*

\* Division of Labor

\* Raja: exercises where (i % 3) == 1

\* Jorge: exercises where (i % 3) == 2

\* Terence: exercises where (i % 3) == 0

\*/

/\*\*

\* Pre-homework table modification

\*

\* Before starting this homework assignment, I want you to change the data

\* type of two fields in one table – the EmployeePay table. In the

\* EmployeePay table, I want the DateStartPay and DateEnd fields to

\* be datetime data types rather than date data types. The easiest way

\* to complete this change is to just drop the table, create it, and

\* repopulate it. Some of the DateEnd fields need to have a time in them,

\* so it seemed most expedient if I just created a SQL script file that

\* would accomplish those goals for you. The file is called

\* BuildEmployeePayHW5.sql and is located on the K: drive in the

\* IS475\CutGlassS15 folder.

\*/

-- Just used the given file.

/\*\*

\* Exercise 01

\*

\* Modify the Job table:

\*/

-- Part a

-- Add a column to the table for DateDue. It should be a date data type.

ALTER TABLE

[Job]

ADD

[DateDue] [date] --NOT NULL?

;

-- Part b

-- UPDATE the new column with the DateDue data given below. I list the

-- JobID and the new DateDue for each job. If using the UPDATE command,

-- you will have to write 10 UPDATE statements.

UPDATE

[Job]

SET

[DateDue] = '2015-01-15'

WHERE

[JobID] = 16885

;

UPDATE

[Job]

SET

[DateDue] = '2014-08-01'

WHERE

[JobID] = 32687

;

UPDATE

[Job]

SET

[DateDue] = '2013-08-01'

WHERE

[JobID] = 55841

;

UPDATE

[Job]

SET

[DateDue] = '2013-10-01'

WHERE

[JobID] = 55873

;

UPDATE

[Job]

SET

[DateDue] = '2013-09-15'

WHERE

[JobID] = 55878

;

UPDATE

[Job]

SET

[DateDue] = '2014-11-20'

WHERE

[JobID] = 62254

;

UPDATE

[Job]

SET

[DateDue] = '2014-11-18'

WHERE

[JobID] = 62257

;

UPDATE

[Job]

SET

[DateDue] = '2015-02-06'

WHERE

[JobID] = 78431

;

UPDATE

[Job]

SET

[DateDue] = '2014-04-01'

WHERE

[JobID] = 91584

;

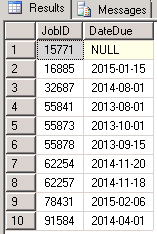
--uncomment to run select query to see results

SELECT

[JobID],

[DateDue]

from [JOB] ;



/\*\*

\* Exercise 02

\*

\* List all jobs with a DateDue in the previous year (i.e. 2014 if the code

\* is run in 2015). Use GETDATE() to determine the previous year. Calculate

\* the difference in days between the Date a job was accepted and the date

\* it was due.

\*/

SELECT

JobId,

JobName,

ClientName,

ClientZip,

DateAccepted,

DateDue,

DATEDIFF(DAY,DateAccepted,DateDue) 'Days To Complete',

EmpManagerID

From

Job INNER JOIN Client ON

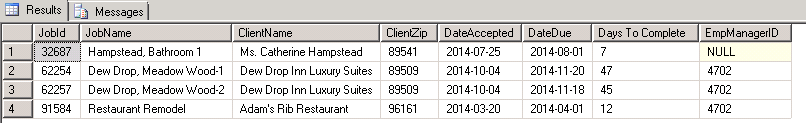
Job.ClientID = Client.ClientID

WHERE

DATEDIFF(YEAR,DateDue,GETDATE()) = 1

ORDER BY

JobID;



/\*\*

\* Exercise 03

\*

\* Modify the query written for question #2 to replace the EmpManagerID

\* in the result table with the name of the employee who was the manager

\* for the purchase order. If the manager is null, display the message

\* "No Manager" in the column. If the manager is not null, then concatenate

\* the FirstName and LastName of the employee into a single column with

\* a space between the first and last names.

\*/

SELECT

[JobID] AS 'JobID',

[JobName] AS 'Job Name',

[ClientName] AS 'Client Name',

[ClientZip] AS 'Client Zip',

CONVERT(varchar(12), [DateAccepted],107 )AS 'Date Accepted',

CONVERT ( varchar(12), [DateDue], 107) AS 'Date Due',

DATEDIFF(DAY, ISNULL([DateAccepted], GETDATE()), ISNULL([DateDue], GETDATE()))

AS 'Days To Complete',

CASE

WHEN [EmpManagerId] IS NULL

THEN 'No Manager'

ELSE

[Employee].[FirstName] + ' ' + [Employee].[LastName]

END AS 'Manager Name'

FROM

[Job]

INNER JOIN [Client] ON

[Job].[ClientID] = [Client].[ClientID]

LEFT OUTER JOIN [Employee] ON

[Job].[EmpManagerID] = [Employee].[EmpID]

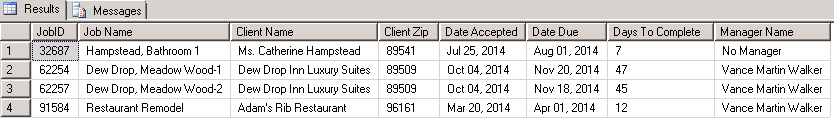
WHERE

DATEDIFF(YEAR, [DateDue], GETDATE()) = 1

ORDER BY

[JobID]

;



/\*\*

\* Exercise 04

\*''

\* Modify the query written for question #3 to include information about

\* the tasks on each of the jobs listed in query #3, as shown below. The

\* data about each task on a job includes the taskID, DateStarted, and

\* DateCompleted for the task. The result table should be sorted by taskID

\* within JobID. JobID should be the primary sort key.

\*/

SELECT

[Job].JobID AS 'JobID',

[JobName] AS 'Job Name',

[ClientName] AS 'Client Name',

[ClientZip] AS 'Client Zip',

CONVERT (varchar(12), [DateAccepted],107 )AS 'Date Accepted',

CONVERT ( varchar(12), [DateDue], 107) AS 'Date Due',

DATEDIFF(DAY, ISNULL([DateAccepted], GETDATE()), ISNULL([DateDue], GETDATE()))

AS 'Days To Complete',

CASE

WHEN [EmpManagerId] IS NULL

THEN 'No Manager'

ELSE

[Employee].[FirstName] + ' ' + [Employee].[LastName]

END AS 'Manager Name',

[JobTask].TaskID As 'taskID',

CONVERT (varchar(12), [JobTask].DateStarted, 107) AS 'DateStarted ',

CONVERT (varchar (12), [JobTask].DateCompleted, 107) AS 'DateCompleted'

FROM

[Job]

INNER

JOIN [Client]

ON [Job].[ClientID] = [Client].[ClientID]

LEFT

OUTER

JOIN [Employee]

ON [Job].[EmpManagerID] = [Employee].[EmpID]

INNER

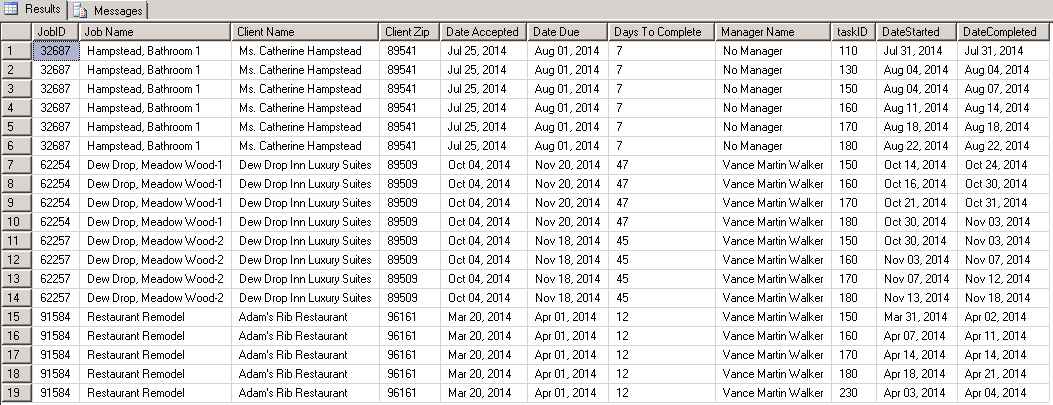
JOIN [JobTask]

ON [Job].JobID = [JobTask].JobID

WHERE

DATEDIFF(YEAR, [DateDue], GETDATE()) = 1

ORDER BY [JobID]



/\*\*

\* Exercise 05

\*

\* Modify the query written for question #4 to remove the JobName and

\* ClientName and include the TaskDescription.

\*/

SELECT

Job.JobId,

Client.ClientZip,

convert(varchar, Job.DateAccepted, 109) 'Date Accepted',

convert(varchar, Job.DateDue, 109) 'Date Due',

DATEDIFF(DAY,Job.DateAccepted,Job.DateDue) 'Days To Complete',

CASE

WHEN Job.EmpManagerID is null

THEN 'No Manager'

ELSE

Employee.FirstName + ' ' + Employee.LastName

END 'Manager Name',

Task.TaskId,

Task.TaskDescription,

convert(varchar,JobTask.DateStarted,109) 'Date Started',

convert(varchar,JobTask.DateCompleted,109) 'Date Completed'

From

Job INNER JOIN Client ON

Job.ClientID = Client.ClientID

LEFT OUTER JOIN Employee ON

Employee.EmpID = Job.EmpManagerID

INNER JOIN JobTask ON

Job.JobID = JobTask.JobID

INNER JOIN Task ON

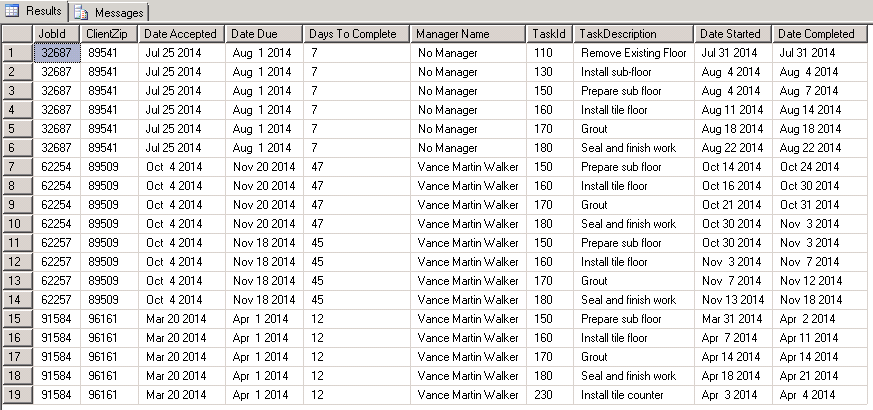
JobTask.TaskID = Task.TaskID

WHERE

DATEDIFF(YEAR,Job.DateDue,GETDATE()) = 1

ORDER BY

JOb.JobID;



/\*\*

\* Exercise 06

\*

\* Final modification. Modify the query written for question #5 to include

\* a self-join with the Job table. The goal of the self-join is to include

\* the JobName of the primary job that is related to the job – I named the

\* JobName associated with the PrimaryJobID as PrimaryJobName in the result

\* table. I used the ISNULL function to put the message "No Primary Job

\* Name" into the PrimaryJobName column.

\*/

SELECT

[Job].[JobID] AS 'JobID',

[Client].[ClientZip] AS 'Client Zip',

CONVERT (varchar(12), [Job].[DateAccepted], 107) AS 'Date Accepted',

CONVERT ( varchar(12), [Job].[DateDue], 107) AS 'Date Due',

ISNULL(

DATEDIFF(

DAY,

ISNULL([Job].[DateAccepted], GETDATE()),

[Job].[DateDue]

),

9999 --Assumed that jobs without deadlines have no practical timeline

) AS 'Days To Complete',

CASE

WHEN [Job].[EmpManagerId] IS NULL

THEN 'No Manager'

ELSE

[Employee].[FirstName] + ' ' + [Employee].[LastName]

END AS 'Manager Name',

ISNULL([PrimaryJob].[JobName], 'No Primary Job Name') AS 'Primary Job Name',

[JobTask].[TaskID] AS 'TaskID',

[Task].[TaskDescription] AS 'Task Description',

CONVERT(varchar(12), [JobTask].[DateStarted], 107) AS 'Date Started',

CONVERT (varchar(12), [JobTask].[DateCompleted], 107) AS 'Date Completed'

FROM

[Job]

INNER JOIN [Client] ON

[Job].[ClientID] = [Client].[ClientID]

LEFT OUTER JOIN [Employee] ON

[Job].[EmpManagerID] = [Employee].[EmpID]

LEFT OUTER JOIN [Job] [PrimaryJob] ON

[Job].[PrimaryJobID] = [PrimaryJob].[JobID],

[JobTask]

INNER JOIN [Task] ON

[JobTask].[TaskID] = [Task].[TaskID]

WHERE

[Job].[JobID] = [JobTask].[JobID] AND

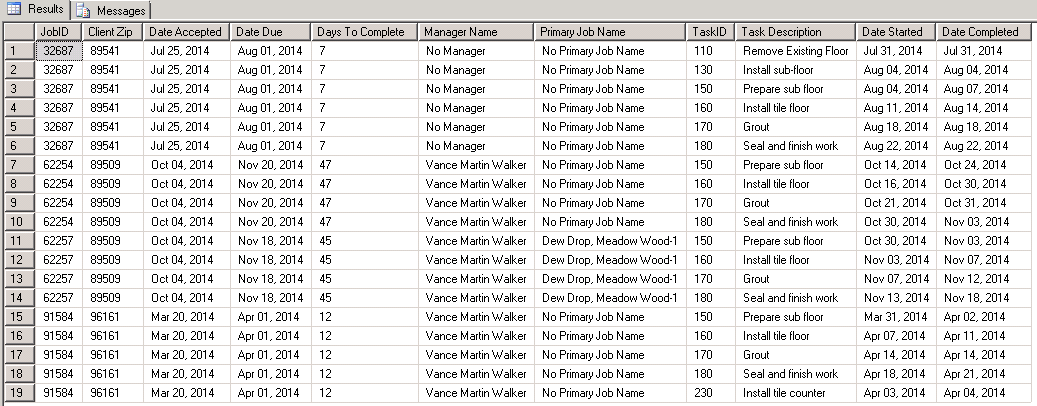
DATEDIFF(YEAR, [Job].[DateDue], GETDATE()) = 1

ORDER BY

[JobID],

[TaskID]

;



/\*\*

\* Exercise 07

\*

\* Time for a new, but similar type of query! List information about all

\* JobTasks in the JobTask table that have a JobTask DateCompleted that

\* is greater than the DateDue for the Job (in the Job table). Sort the

\* output by TaskID within JobID.

\*/

SELECT

[JobTask].JobID AS 'JobID',

[Job].JobName AS 'JobName',

[Client].ClientName AS 'ClientName',

[JobTask].TaskID AS 'TaskID',

[Task].TaskDescription AS 'TaskDescription',

[Job].DateDue AS 'DateDue',

[JobTask].DateCompleted AS 'DateCompleted',

DATEDIFF (DAY, [Job].DateDue,[JobTask].[DateCompleted]) AS 'DaysOverdue'

FROM [JobTask]

INNER

JOIN [Job]

ON [JobTask].JobID = [Job].JobID

INNER

JOIN [Client]

ON [Job].ClientID = [Client].ClientID

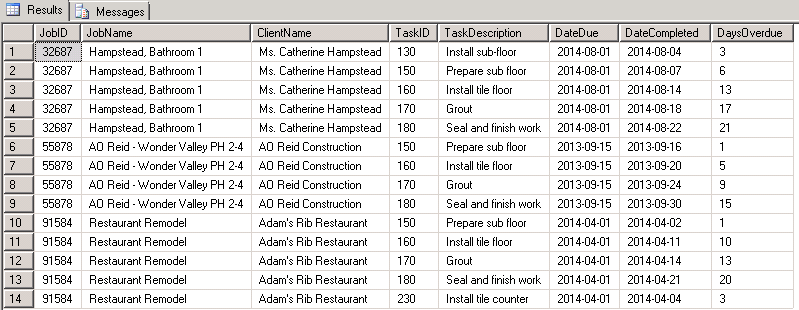
INNER

JOIN [Task]

ON [JobTask].TaskID = [Task].TaskID

WHERE DATEDIFF (DAY, [Job].DateDue,[JobTask].[DateCompleted]) > 0

ORDER BY [JobTask].JobID, [JobTask].TaskID;



/\*\*

\* Exercise 08

\*

\* Modify query #7 so that it also shows the job tasks that haven't been

\* completed yet (DateCompleted is null) as of when your SQL code runs,

\* but are beyond the duedate of the job. I ran this SQL code on 2/18/2015,

\* so the DaysOverdue calculation reflect that current date. Your output

\* should reflect the current date when it is run.

\*/

SELECT

Job.JobID,

Job.JobName,

Client.ClientName,

JobTask.TaskId,

Job.DateDue,

ISNULL(CAST(JobTask.DateCompleted AS varchar),'Not Done') 'Date Completed',

DATEDIFF(DAY,ISNULL(JobTask.DateCompleted,GETDATE()),Job.DateDue) \* -1 'DaysOverDue'

FROM

Job INNER JOIN CLient ON

Job.ClientID = Client.ClientID

INNER JOIN JobTask ON

Job.JobID = JobTask.JobID

INNER JOIN Task ON

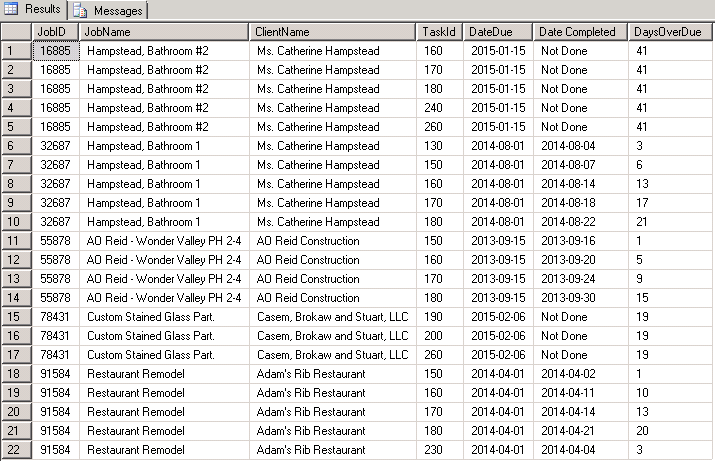
JobTask.taskID = Task.Taskid

WHERE

DATEDIFF(DAY,JobTask.DateCompleted,Job.DateDue) < 0 or JobTask.DateCompleted is null

ORDER BY

JobID;



/\*\*

\* Exercise 09

\*

\* Time for a new query! List the material costs by task for JobID 91584.

\* The TotalCost is the CostPerUOM in the MaterialPurchased table

\* multiplied by the Quantity assigned in the MaterialAssigned table.

\* Cast or CONVERT the TotalCost to a money data type. Sort the result table

\* by materialID within taskID.

\*/

SELECT

[MaterialAssigned].[JobID] AS 'JobID',

[MaterialAssigned].[TaskID] AS 'TaskID',

[MaterialPurchased].[MaterialID] AS 'MaterialID',

[MaterialAssigned].[Quantity] AS 'Quantity',

[MaterialPurchased].[CostPerUOM] AS 'Cost Per UOM',

CAST(

([MaterialPurchased].[CostPerUOM] \* [MaterialAssigned].[Quantity])

AS MONEY)

AS 'Actual Cost'

FROM

[MaterialAssigned]

INNER JOIN [MaterialPurchased] ON

[MaterialAssigned].[POID] = [MaterialPurchased].[POID]

WHERE

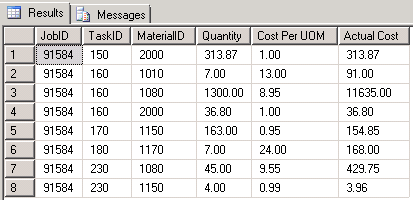
[MaterialAssigned].[JobID] = 91584

ORDER BY

[MaterialAssigned].[TaskID],

[MaterialPurchased].[MaterialID]

;



/\*\*

\* Exercise 10

\*

\* Modify query #9 to summarize the costs of materials by taskID.

\*/

SELECT

[MaterialAssigned].[JobID] AS 'JobID',

[MaterialAssigned].[TaskID] AS 'TaskID',

SUM([MaterialAssigned].[Quantity]) AS 'Actual Quantity',

CAST(

SUM([MaterialPurchased].[CostPerUOM] \* [MaterialAssigned].[Quantity])

AS MONEY)

AS 'Actual Cost'

FROM

[MaterialAssigned]

INNER JOIN [MaterialPurchased] ON

[MaterialAssigned].[POID] = [MaterialPurchased].[POID]

WHERE

[MaterialAssigned].[JobID] = 91584

GROUP BY

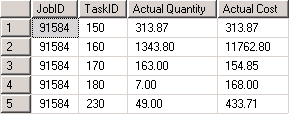
[MaterialAssigned].[JobID],

[MaterialAssigned].[TaskID]

ORDER BY

[MaterialAssigned].[TaskID]

;



/\*\*

\* Exercise 11

\*

\* Modify query #10 to compare the actual material costs generated in

\* that query to the estimated material costs in the JobTask table.

\* Calculate the difference between the estimated cost of materials for

\* a task to the actual cost for the materials for a task.

\*/

SELECT

[MaterialAssigned].[JobID] AS 'JobID',

[MaterialAssigned].[TaskID] AS 'TaskID',

MIN([JobTask].[DateCompleted]) AS 'Date Completed',

([JobTask].[EstMaterialCost]) AS 'Estimated Material Cost',

CAST(

SUM([MaterialPurchased].[CostPerUOM] \* [MaterialAssigned].[Quantity])

AS MONEY)

AS 'Actual Cost',

CAST(

([JobTask].[EstMaterialCost]) -

SUM([MaterialPurchased].[CostPerUOM] \* [MaterialAssigned].[Quantity])

AS MONEY)

AS 'Difference Estimated/Actual'

FROM

[MaterialAssigned]

INNER JOIN [MaterialPurchased] ON

[MaterialAssigned].[POID] = [MaterialPurchased].[POID]

INNER JOIN [JobTask] ON

[MaterialAssigned].[JobID] = [JobTask].[JobID] AND

[MaterialAssigned].[TaskID] = [JobTask].[TaskID]

WHERE

[MaterialAssigned].[JobID] = 91584

GROUP BY

[MaterialAssigned].[JobID],

[MaterialAssigned].[TaskID],

[JobTask].[JobID],

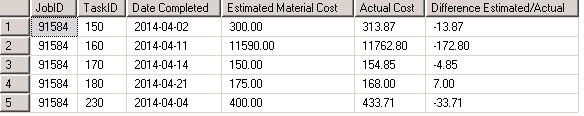
[JobTask].[TaskID],

[JobTask].[EstMaterialCost]

ORDER BY

[MaterialAssigned].[TaskID]

;



/\*\*

\* Exercise 12

\*

\* Modify query #11 to summarize the estimated material cost and actual

\* material cost for the entire job. This may prove to be difficult because of

\* the way that joins work. The MaterialAssigned table is the child of the

\* JobTask table in the relationship between the two. Thus, a join between

\* the two will always produce a result table with the number of rows in

\* the child table (MaterialAssigned). This may yield an incorrect result

\* if you are trying to sum a quantity that is in the parent table, while

\* also summing a quantity that is in the child table. Hint: Use a

\* sub-query in the SELECT list to get the correct sum of the

\* EstMaterialCost in the parent table (JobTask).

\*/

GO

CREATE VIEW v\_EstimatedMaterialCostSums AS

SELECT

[JobTask].[JobID] AS 'JobID',

SUM([JobTask].[EstMaterialCost]) AS 'EstimatedMaterialCost',

SUM([JobTask].[EstLaborCost]) AS 'EstimatedLaborCost'

FROM

[JobTask]

GROUP BY

[JobTask].[JobID]

;

GO

CREATE VIEW v\_ActualMaterialCostSums AS

SELECT

[MaterialAssigned].[JobID] AS 'JobID',

SUM(

[MaterialAssigned].[Quantity] \*

[MaterialPurchased].[CostPerUOM]

) AS 'ActualMaterialCost'

FROM

[MaterialAssigned]

INNER JOIN [MaterialPurchased] ON

[MaterialAssigned].[POID] = [MaterialPurchased].[POID]

GROUP BY

[MaterialAssigned].[JobID]

;

GO

SELECT

[v\_EstimatedMaterialCostSums].[JobID] AS 'JobID',

CAST(

SUM([v\_EstimatedMaterialCostSums].[EstimatedMaterialCost])

AS MONEY

) AS 'Estimated Material Cost',

CAST(

SUM([v\_ActualMaterialCostSums].[ActualMaterialCost])

AS MONEY

) AS 'Actual Material Cost',

CAST(

SUM([v\_EstimatedMaterialCostSums].[EstimatedMaterialCost]) -

SUM([v\_ActualMaterialCostSums].[ActualMaterialCost])

AS MONEY

) AS 'Difference Estimated/Actual'

FROM

[v\_EstimatedMaterialCostSums]

INNER JOIN [v\_ActualMaterialCostSums] ON

[v\_EstimatedMaterialCostSums].[JobID] = [v\_ActualMaterialCostSums].[JobID]

WHERE

[v\_EstimatedMaterialCostSums].[JobID] = 91584

GROUP BY

[v\_EstimatedMaterialCostSums].[JobID]

ORDER BY

[v\_EstimatedMaterialCostSums].[JobID]

;

DROP VIEW v\_EstimatedMaterialCostSums;

DROP VIEW v\_ActualMaterialCostSums;



/\*\*

\* Exercise 13

\*

\* Time for a new query! Summarize the total amount of time worked for

\* each employee in the database. In addition to summing the HoursWorked,

\* also count the number of TimeSheets for each employee.

\*/

SELECT

Employee.EmpID,

Employee.LastName + ',' + Employee.firstName 'Employee Name',

ISNULL(SUM(Timesheet.HoursWorked),0) 'TotalHoursWorked',

(SELECT COUNT(EmpID) FROM TimeSheet WHERE Employee.EmpID = TimeSheet.EmpID) 'Number OF Timesheets'

From

Employee LEFT OUTER JOIN TIMESHEET ON

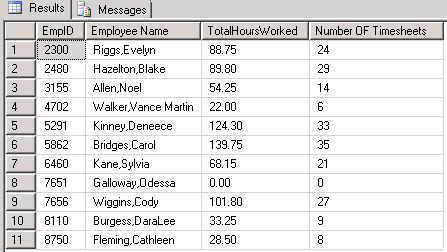
Employee.EmpID = TimeSheet.EmpID

GROUP BY

Employee.EmpID,

Employee.LastName,

Employee.FirstName



/\*\*

\* Exercise 14

\*

\* Another new query! Summarize all the time worked in the TimeSheet table

\* by task.

\*/

SELECT

Task.TaskID,

Task.TaskDescription,

ISNULL(SUM(TimeSheet.HoursWorked),0) 'Total Hours'

FROM

Task LEFT OUTER JOIN TimeSheet ON

Task.taskID = TImeSheet.taskID

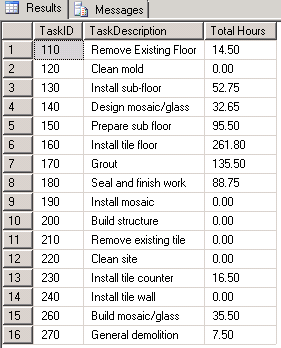
GROUP BY

Task.TaskID,

Task.TaskDescription

ORDER BY

Task.TaskID



/\*\*

\* Exercise 15

\*

\* Another new query (important for HW6). Eventually, we want to be able

\* to compare the actual number of hours worked and the actual cost of

\* those hours to the estimated hours and estimated labor cost. However,

\* this data is spread over a number of related tables in the database,

\* making it necessary to join data and use sub-queries. Right now, all

\* we want to do is figure out the actual hours worked and the actual cost

\* of those hours for every row in the TimeSheet table where the jobID is

\* NOT NULL. Sort the result table by EmpID.

\*

\* The hourly pay rate for an employee is stored in the EmployeePay table,

\* and that payrate can change over time, so we need to determine the

\* correct pay rate for HoursWorked based on the date that work was performed

\* in the TimeSheet table and the time in the EmployeePay table. I recommend

\* that you look back at HW#4, question #10 to remember how to locate the

\* correct HourlyPayRate for a given period of time for a given employee.

\* There should be 199 rows in the result table, so I have broken it up

\* into multiple snips but included all rows for your reference. You do

\* not have to include all rows on your output. As shown and sorted below,

\* I'd like to see rows 1-5, 125-135, 180-184 on your output so that I can

\* verify your results.

\*/

SELECT

TimeSheet.EmpID AS 'EmpID',

Employee.LastName + ', ' + Employee.FirstName AS 'Employee Name',

TimeSheet.TaskID AS 'TaskID',

TimeSheet.JobID AS 'JobID',

TimeSheet.StartWork AS 'StartWork',

TimeSheet.HoursWorked AS 'HoursWorked',

EmployeePay.HourlyPayRate AS 'HourlyPay',

TimeSheet.HoursWorked \* EmployeePay.HourlyPayRate AS 'LaborCost'

FROM

TimeSheet

INNER JOIN EmployeePay ON

TimeSheet.EmpID = EmployeePay.EmpID

INNER JOIN Employee ON

TimeSheet.EmpID = Employee.EmpID

WHERE

EmployeePay.DateStartPay <= TimeSheet.StartWork AND

TimeSheet.StartWork < ISNULL(EmployeePay.DateEnd, GETDATE()) AND

TimeSheet.JobID IS NOT NULL

;

