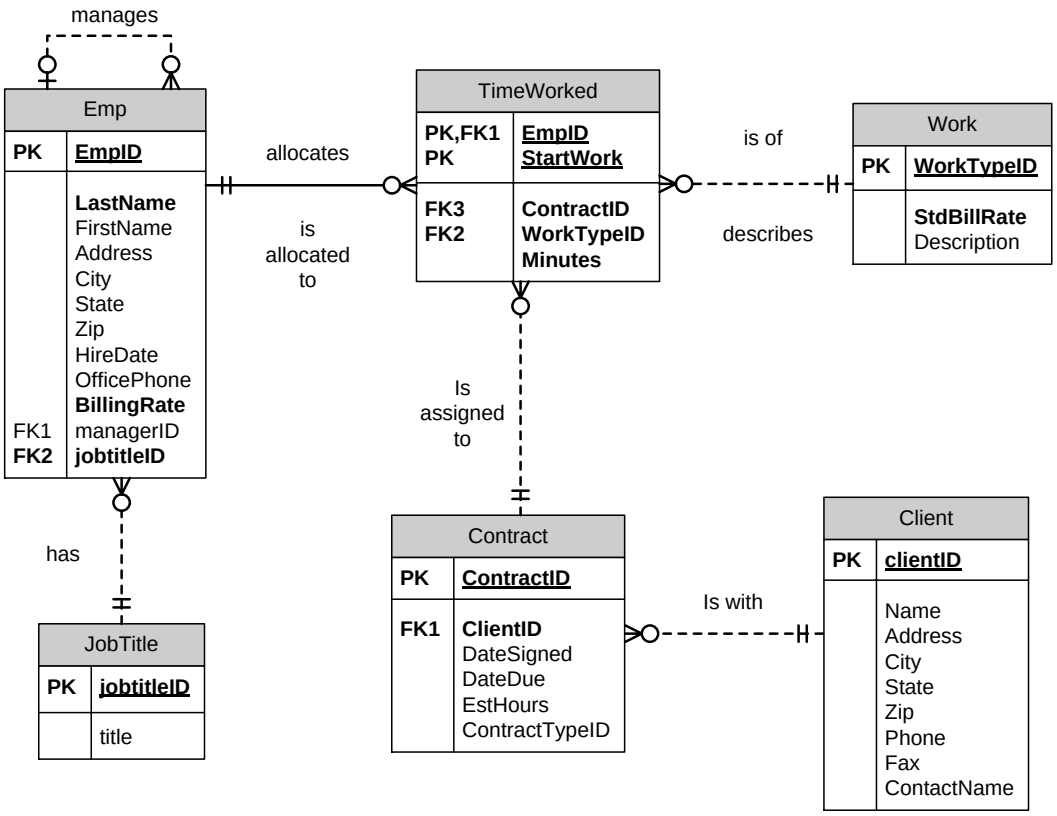
IS475/675 SQL Join Lab – 03/08/2016

At the start of class do the following: (1) login to SQL Server; (2) read this page; and (3) complete Task 1 step (1) on the next page.



The database above is for a computer consulting company. The consulting company requires all employees to record time spent working doing a particular type of work for a given contract. This is the database where employee time records are stored and related to work for a client's contract.

The "minutes" attribute in the TIMEWORKED entity is the number of minutes that an employee spent doing work for a given contract. For example, if an employee worked 15 minutes for a given contract, then the minutes would be 15. If an employee worked an hour for a contract, then the minutes would be 60. If an employee worked 8 hours on a given day for just a single contract, then the minutes would be 480.

The WORK table contains a standardized description of the types of work consultants perform in the organization with a standard billing rate for each type of work. For example, a type of work performed at this company is "JAVA programming" and the standard billing rate for that type of work is $75/hour. That information is stored in the WORK table.

The CONTRACT table stores data about contracts from clients. The "EstHours" attribute in this table stores the total amount of time (in hours) that the computer consulting company plans to work in total on a given contract.

Referential Integrity is maintained in this database, except between the Emp and JobTitle tables. It is possible that an employee has a job title ID that is not in the JobTitle table.

Task 1. Please do step (1) below at the start of class. We will do the rest of the steps together during class.

(1) Create and populate the tables by executing kdrive:\is475\LabFiles\Create03-8Lab.sql. After executing the script file, write down the number of rows for each table in the database. Write this information directly on the ERD shown on the previous page. Which tables are "parent" tables and which are "child" tables? Indicate the parent and child for each relationship on the diagram directly on the ERD.

(2) Let's learn about referential integrity. We are maintaining referential integrity between all relationships except for the Emp and JobTitle tables. Type the following code:

SQL CODE	What does it show? (write your notes)

`SELECT *`
`FROM labtimeworked`
`WHERE empid NOT IN`
`(SELECT empid`
`FROM labemp);`	

`SELECT *`
`FROM labemp`
`WHERE empid NOT IN`
`(SELECT empid`
`FROM labtimeworked);`	

`SELECT *`
`FROM labemp`
`WHERE jobtitleid NOT IN`
`(SELECT jobtitleid`
`FROM labjobtitle);`	

`SELECT *`
`FROM labjobtitle`
`WHERE jobtitleid NOT IN`
`(SELECT jobtitleid`
`FROM labemp);`	

(3) Let's learn about the syntax of joins, working with multiple tables, and the difference between INNER and OUTER joins.

An INNER join combines two tables (multiplies one table by the other) and includes only the rows where the primary key of the parent table is equal to the foreign key of the child table.

Type this code:

```
SELECT      *
FROM        labtimeworked
INNER JOIN  labemp
ON          labemp.empid = labtimeworked.empid
ORDER BY    labemp.empid
```

Modify the code to look like this:

```
SELECT      labemp.empid,
            labemp.lastname,
            labemp.firstname,
            labtimeworked.startwork,
            labtimeworked.minutes,
            labemp.billingrate
FROM        labtimeworked
INNER JOIN  labemp
ON          labemp.empid = labtimeworked.empid
ORDER BY    labemp.empid
```

Modify the code to look like this to see how to abbreviate table names:

```
SELECT        emp.empid,
              emp.lastname,
              emp.firstname,
              tw.startwork,
              tw.minutes,
              emp.billingrate
FROM          labtimeworked tw
INNER JOIN    labemp emp
ON            emp.empid = tw.empid
ORDER BY      emp.empid
```

Modify the code to look like this to understand there is no difference between an INNER and OUTER join when maintaining referential integrity and pointing the OUTER join to the child table (in this case, the child table is labTimeWorked, which is on the LEFT side of the join):

```
SELECT        emp.empid,
              emp.lastname,
              emp.firstname,
              tw.startwork,
              tw.minutes,
              emp.billingrate
FROM          labtimeworked tw
LEFT OUTER JOIN     labemp emp
ON            emp.empid = tw.empid
ORDER BY      emp.empid
```

Modify the code to look like this to understand there is a difference between an INNER and OUTER join when maintaining referential integrity and pointing the OUTER join to the parent table (in this case, the parent table is labEmp, which is on the right side of the join). The result table now includes the two employees who do not have any work recorded in the TimeWorked table.

```
SELECT        emp.empid,
              emp.lastname,
              emp.firstname,
              tw.startwork,
              tw.minutes,
              emp.billingrate
FROM          labtimeworked tw
RIGHT OUTER JOIN    labemp emp
ON            emp.empid = tw.empid
ORDER BY      emp.empid
```

(4) Let's test your knowledge of joining tables by doing a "conceptual" problem. I want to join together the TimeWorked table and the Contract table. There are 36 rows in the TimeWorked table and 6 rows in the Contract table. We are maintaining referential integrity between these two tables. When these two tables are joined with an INNER join, how many rows will be in the result table? You should not need to run any code to figure out this problem, but here is the code to join these two tables:

```
SELECT        *
FROM          labTimeWorked
INNER JOIN    labContract
ON            labtimeworked.contractID = labcontract.contractID
```

(5) Another conceptual problem to test your knowledge. Let's say that there are 3 rows in the Contract table that have no work recorded in the TimeWorked table. Let's say that we do an OUTER join to join the two tables together and we point to the Contract table. How many rows will be in the result table? The code is at the top of the next page.

```
SELECT        *
FROM          labTimeWorked
RIGHT OUTER JOIN    labContract
ON            labtimeworked.contractID = labcontract.contractID
```

(6) Another conceptual problem to test your knowledge. Let's say that the constraints discussed previously are still true – we are maintaining referential integrity and there are 3 rows in the Contract table that have no work recorded in the TimeWorked table. Let's say that we do an OUTER join to join the two tables together and we point to the TimeWorked table. How many rows will be in the result table? Here is the code:

```
SELECT        *
FROM          labTimeWorked
LEFT OUTER JOIN     labContract
ON            labtimeworked.contractID = labcontract.contractID
```

(7) A final conceptual problem to test your knowledge. Let's say that the constraints discussed previously are still true – we are maintaining referential integrity and there are 3 rows in the Contract table that have no work recorded in the TimeWorked table. Let's say that we do an OUTER join to join the two tables together and we point to both tables. This is called a FULL OUTER JOIN. How many rows will be in the result table? Here is the code:

```
SELECT        *
FROM          labTimeWorked
FULL OUTER JOIN     labContract
ON            labtimeworked.contractID = labcontract.contractID
```

(8) Let's go back to the INNER join code and add another table. We are going to add the JobTitle table so that we can see a person's job title along with the data available in the Emp and TimeWorked tables. Modify the code to look like this to see how to abbreviate table names:

```
SELECT        emp.empid,
              emp.lastname,
              emp.firstname,
              jt.title JobTitle,
              tw.startwork,
              tw.minutes,
              emp.billingrate
FROM          labtimeworked tw
INNER JOIN    labemp emp
ON            emp.empid = tw.empid
LEFT OUTER JOIN labjobtitle jt
ON            jt.jobtitleid = emp.jobtitleid
ORDER BY      emp.empid
```

(9) Let's make the job title say "**Missing**" if it is null.

```
SELECT        emp.empid,
              emp.lastname,
              emp.firstname,
              ISNULL(jt.title, '**Missing**') JobTitle,
              tw.startwork,
              tw.minutes,
              emp.billingrate
FROM          labtimeworked tw
INNER JOIN    labemp emp
ON            emp.empid = tw.empid
LEFT OUTER JOIN labjobtitle jt
ON            jt.jobtitleid = emp.jobtitleid
ORDER BY      emp.empid
```

Time to work independently. The primary goal of this lab is to learn how to join tables, and to understand when to do an INNER join vs. an OUTER join. The secondary goal is to reinforce your knowledge of GROUP BY and sub-queries.

Task 2. Practice joining tables. List all the rows in the EMP table, along with the title for each employee from the JOBTITLE table. Since referential integrity is not maintained between the EMP and JOBTITLE tables, should you use an inner or outer join? Sort the result table by EmpID. Remember that all tables in this database have the prefix "lab". The result table is provided at the top of the next page.

	empid	lastname	firstname	title
1	2412	Perez	Martina	SAP Analyst
2	3411	Tristan	Elliott	NULL
3	3424	Polanski	Charles	Graphics Designer
4	4522	Jenkins	Martin	Manager
5	5633	Flanders	Janice	Business Analyst
6	7369	Chu	Joyce	Business Analyst
7	7715	Kendall	Brent	Database Designer
8	7819	Martinson	Crandall	Manager

Task 3. Enhance the query created for Task #2 to display a message in the Title column if the result is null. The result table is on the top of the next page.

	empid	lastname	firstname	JobTitle
1	2412	Perez	Martina	SAP Analyst
2	3411	Tristan	Elliott	**Missing**
3	3424	Polanski	Charles	Graphics Designer
4	4522	Jenkins	Martin	Manager
5	5633	Flanders	Janice	Business Analyst
6	7369	Chu	Joyce	Business Analyst
7	7715	Kendall	Brent	Database Designer
8	7819	Martinson	Crandall	Manager

Task 4. Enhance the query created for Task #3 to list all the rows in the TIMEWORKED table along with the name of the employee who did the work. Convert the minutes to hours (in other words, divide by 60). Sort the result table by the empid in the TIMEWORKED table. Display only those employees in the EMP table that have completed work in the TIMEWORKED table. The result table is provided on the next page.

	empid	lastname	firstname	JobTitle	startwork	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	3
4	3411	Tristan	Elliott	**Missing**	2016-02-12 08:00:00.000	11
5	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	8
6	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	2
7	4522	Jenkins	Martin	Manager	2016-01-10 08:00:00.000	9
8	4522	Jenkins	Martin	Manager	2016-01-11 08:00:00.000	10
9	4522	Jenkins	Martin	Manager	2016-01-12 08:00:00.000	10
10	4522	Jenkins	Martin	Manager	2016-01-13 08:00:00.000	10
11	4522	Jenkins	Martin	Manager	2016-01-14 08:00:00.000	10
12	4522	Jenkins	Martin	Manager	2016-02-27 08:00:00.000	3
13	4522	Jenkins	Martin	Manager	2016-02-27 11:00:00.000	3
14	4522	Jenkins	Martin	Manager	2016-02-27 15:00:00.000	2
15	4522	Jenkins	Martin	Manager	2016-02-28 08:00:00.000	8
16	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	2
17	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	12
18	5633	Flanders	Janice	Business Analyst	2016-01-02 08:00:00.000	8
19	5633	Flanders	Janice	Business Analyst	2016-01-03 08:00:00.000	10
20	5633	Flanders	Janice	Business Analyst	2016-01-04 08:00:00.000	10
21	5633	Flanders	Janice	Business Analyst	2016-01-05 08:00:00.000	10
22	5633	Flanders	Janice	Business Analyst	2016-02-19 08:00:00.000	7
23	7369	Chu	Joyce	Business Analyst	2016-01-10 08:00:00.000	9
24	7369	Chu	Joyce	Business Analyst	2016-02-10 08:00:00.000	8
25	7369	Chu	Joyce	Business Analyst	2016-02-10 15:00:00.000	2
26	7369	Chu	Joyce	Business Analyst	2016-02-11 08:00:00.000	10
27	7369	Chu	Joyce	Business Analyst	2016-02-12 08:00:00.000	10
28	7369	Chu	Joyce	Business Analyst	2016-02-13 08:00:00.000	10
29	7369	Chu	Joyce	Business Analyst	2016-02-14 08:00:00.000	10
30	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	2
31	7819	Martinson	Crandall	Manager	2016-01-15 09:00:00.000	5
32	7819	Martinson	Crandall	Manager	2016-01-16 09:00:00.000	1
33	7819	Martinson	Crandall	Manager	2016-01-28 08:00:00.000	2
34	7819	Martinson	Crandall	Manager	2016-02-06 08:00:00.000	1
35	7819	Martinson	Crandall	Manager	2016-02-06 16:00:00.000	5
36	7819	Martinson	Crandall	Manager	2016-02-10 09:00:00.000	2

Task 5. Modify the query created for Task #4 to include all the employees in the EMP table, whether or not the employee completed any work in the TIMEWORKED table. The result table is provided below. Note that there are two new rows in the result table – row #7 for Polanski and row #32 for Kendall. These two employees do not have any related rows in the TIMEWORKED table, so their data from that table is NULL.

	empid	lastname	firstname	JobTitle	startwork	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	3
4	3411	Tristan	Elliott	**Missing**	2016-02-12 08:00:00.000	11
5	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	8
6	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	2
7	3424	Polanski	Charles	Graphics Designer	NULL	NULL
8	4522	Jenkins	Martin	Manager	2016-01-10 08:00:00.000	9
9	4522	Jenkins	Martin	Manager	2016-01-11 08:00:00.000	10
10	4522	Jenkins	Martin	Manager	2016-01-12 08:00:00.000	10
11	4522	Jenkins	Martin	Manager	2016-01-13 08:00:00.000	10
12	4522	Jenkins	Martin	Manager	2016-01-14 08:00:00.000	10
13	4522	Jenkins	Martin	Manager	2016-02-27 08:00:00.000	3
14	4522	Jenkins	Martin	Manager	2016-02-27 11:00:00.000	3
15	4522	Jenkins	Martin	Manager	2016-02-27 15:00:00.000	2
16	4522	Jenkins	Martin	Manager	2016-02-28 08:00:00.000	8
17	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	2
18	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	12
19	5633	Flanders	Janice	Business Analyst	2016-01-02 08:00:00.000	8
20	5633	Flanders	Janice	Business Analyst	2016-01-03 08:00:00.000	10
21	5633	Flanders	Janice	Business Analyst	2016-01-04 08:00:00.000	10
22	5633	Flanders	Janice	Business Analyst	2016-01-05 08:00:00.000	10
23	5633	Flanders	Janice	Business Analyst	2016-02-19 08:00:00.000	7
24	7369	Chu	Joyce	Business Analyst	2016-01-10 08:00:00.000	9
25	7369	Chu	Joyce	Business Analyst	2016-02-10 08:00:00.000	8
26	7369	Chu	Joyce	Business Analyst	2016-02-10 15:00:00.000	2
27	7369	Chu	Joyce	Business Analyst	2016-02-11 08:00:00.000	10
28	7369	Chu	Joyce	Business Analyst	2016-02-12 08:00:00.000	10
29	7369	Chu	Joyce	Business Analyst	2016-02-13 08:00:00.000	10
30	7369	Chu	Joyce	Business Analyst	2016-02-14 08:00:00.000	10
31	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	2
32	7715	Kendall	Brent	Database Desig...	NULL	NULL
33	7819	Martins...	Crandall	Manager	2016-01-15 09:00:00.000	5
34	7819	Martins...	Crandall	Manager	2016-01-16 09:00:00.000	1
35	7819	Martins...	Crandall	Manager	2016-01-28 08:00:00.000	2
36	7819	Martins...	Crandall	Manager	2016-02-06 08:00:00.000	1
37	7819	Martins...	Crandall	Manager	2016-02-06 16:00:00.000	5
38	7819	Martins...	Crandall	Manager	2016-02-10 09:00:00.000	2

Task 6. Modify the query created for Task #5 to display only those rows from the TIMEWORKED table that had a startwork date in March of the current year. The result table is shown below.

	empid	lastname	firstname	JobTitle	startwork	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	3
4	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	8
5	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	2
6	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	2
7	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	12
8	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	2

Task 7. Modify the query created for Task #6 to include the description of the type of work performed from the WORK table. The result table is shown below.

	empid	lastname	firstname	JobTitle	startwork	description	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	Statistical Programming	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	Java Programming	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	Statistical Programming	3
4	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	Java Programming	8
5	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	Statistical Programming	2
6	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	Java Programming	2
7	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	Vendor Investigation/Procurement	12
8	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	Java Programming	2

Task 8. Modify the query created for Task #7 to include the contractID from the TIMEWORKED table and the date that the contract is due from the CONTRACT table. The result table is shown below.

	empid	lastname	firstname	JobTitle	startwork	description	contractid	DateDue	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	Statistical Programming	777	Apr 15, 2016	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	Java Programming	444	Apr 01, 2016	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	Statistical Programming	444	Apr 01, 2016	3
4	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	Java Programming	444	Apr 01, 2016	8
5	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	Statistical Programming	444	Apr 01, 2016	2
6	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	Java Programming	777	Apr 15, 2016	2
7	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	Vendor Investigation/Procurement	444	Apr 01, 2016	12
8	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	Java Programming	777	Apr 15, 2016	2

Task 9. Modify the query created for Task #8 to include the name of the client (available in the CLIENT table) for the contract for whom the employee was doing the work. The result table is shown below.

	empid	lastname	firstname	JobTitle	startwork	description	contractid	DateDue	name	HoursWorked
1	2412	Perez	Martina	SAP Analyst	2016-03-03 08:00:00.000	Statistical Programming	777	Apr 15, 2016	Crestwave Supply Co.	8
2	2412	Perez	Martina	SAP Analyst	2016-03-08 08:00:00.000	Java Programming	444	Apr 01, 2016	Cancer Research Society	8
3	2412	Perez	Martina	SAP Analyst	2016-03-09 14:00:00.000	Statistical Programming	444	Apr 01, 2016	Cancer Research Society	3
4	3411	Tristan	Elliott	**Missing**	2016-03-08 08:00:00.000	Java Programming	444	Apr 01, 2016	Cancer Research Society	8
5	3411	Tristan	Elliott	**Missing**	2016-03-08 18:00:00.000	Statistical Programming	444	Apr 01, 2016	Cancer Research Society	2
6	4522	Jenkins	Martin	Manager	2016-03-02 08:00:00.000	Java Programming	777	Apr 15, 2016	Crestwave Supply Co.	2
7	4522	Jenkins	Martin	Manager	2016-03-09 08:00:00.000	Vendor Investigation/Procurement	444	Apr 01, 2016	Cancer Research Society	12
8	7369	Chu	Joyce	Business Analyst	2016-03-06 08:00:00.000	Java Programming	777	Apr 15, 2016	Crestwave Supply Co.	2

Task 10. Completely New query! Aggregating/Grouping and one join. Summarize the amount of work in the TimeWorked table by employee. To accomplish this task, you need to use the Emp and TimeWorked tables. You also need to use the GROUP BY statement and sum the hours worked calculation. Result table:

	empid	lastname	TotalHoursWorked
1	2412	Perez	19
2	3411	Tristan	21
3	4522	Jenkins	79
4	5633	Flanders	45
5	7369	Chu	61
6	7819	Martinson	16

Task 11. Modify the query created for task #10 to include all employees, whether or not they have any matching rows in the TIMEWORKED table. If an employee has a null value for TotalHoursWorked, replace it with a zero. This query requires all rows in the TIMEWORKED table and all rows in the EMP table. The result table is on the top of the next page.

	empid	lastname	TotalHoursWorked
1	2412	Perez	19
2	3411	Tristan	21
3	3424	Polanski	NULL
4	4522	Jenkins	79
5	5633	Flanders	45
6	7369	Chu	61
7	7715	Kendall	NULL
8	7819	Martinson	16

Task 12. Modify the query created for task #11 to include some additional aggregate values. Count the number of rows in the TimeWorked table that are associated with an employee, and also calculate the average hours worked per row. Result table:

	empid	lastname	CountOfRows	AverageHoursPerRow	TotalHoursWorked
1	2412	Perez	3	6	19
2	3411	Tristan	3	7	21
3	3424	Polanski	0	NULL	NULL
4	4522	Jenkins	11	7	79
5	5633	Flanders	5	9	45
6	7369	Chu	8	7	61
7	7715	Kendall	0	NULL	NULL
8	7819	Martinson	6	2	16

Task 13. Modify the query created for task #12 to replace the NULL values with zeroes. Result table:

	empid	lastname	CountOfRows	AverageHoursPerRow	TotalHoursWorked
1	2412	Perez	3	6	19
2	3411	Tristan	3	7	21
3	3424	Polanski	0	0	0
4	4522	Jenkins	11	7	79
5	5633	Flanders	5	9	45
6	7369	Chu	8	7	61
7	7715	Kendall	0	0	0
8	7819	Martinson	6	2	16

Task 14. Completely new query! Find the employee who has the highest billing rate. Don't just display the highest billing rate, display the name of the employee who has the highest billing rate, along with that employee's billing rate. Result table:

	empid	EmployeeName	billingrate
1	3411	Tristan, E.	225.00

Task 15. Enhance the query created for Task #14 to include the employee's job title. If the job title is null, then it should say "missing". Result table:

	empid	EmployeeName	JobTitle	billingrate
1	3411	Tristan, E.	**Missing**	225.00