CIT230 FINAL DATABASE PROJECT

QUOTE database for Ellis Wyatt

Your Name: Jenya Lestina

Course: CIT230 – First Summer 2018

Professor: Nancy S. Grant, Ed.D.

**NARRATIVE OF THE PURPOSE OF THE CREATED DATABASE**

My mother is part-owner of a business called Ellis Wyatt, which provides roofing and other home improvement contracting services. For my last class, in Java, I did my project on creating a Java app for the staff to use to create quotes, so for my SQL project, I figured I would create a database to store and manage the resulting data. Hopefully after both of these classes, I will find a way to merge the two together into a single, working system.

**Database name:** EWDB.mdf

**Tables:**

* dbo.Customer
* dbo.Quote
* dbo.WorkSite

**A LISTING OF ALL OF THE DATABASE TABLE STRUCTURES**

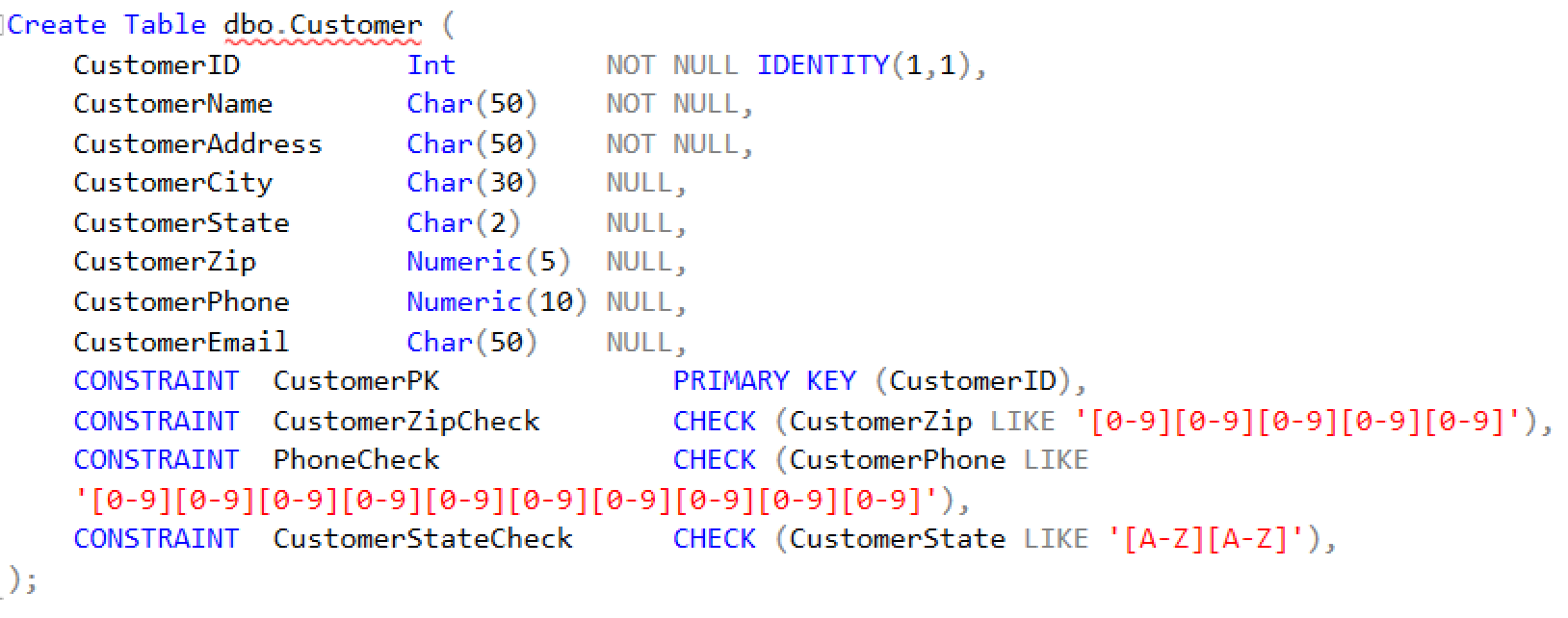
Each table is listed along with a description of its purpose and a picture of the code used to create it.

**Customer**

This table stores each customer of Ellis Wyatt. The primary key, CustomerID, uniquely identifies each customer and allows for the easy integration of the Customer table with each of the others. This table also includes other information about each customer, including full name and address.

The constraints used to validate the data in this table are:

* Zip must be the US standard 5 numeric digits
* State must be two capital letters
* Phone number must be 10 digits with no spaces, dashes, etc.

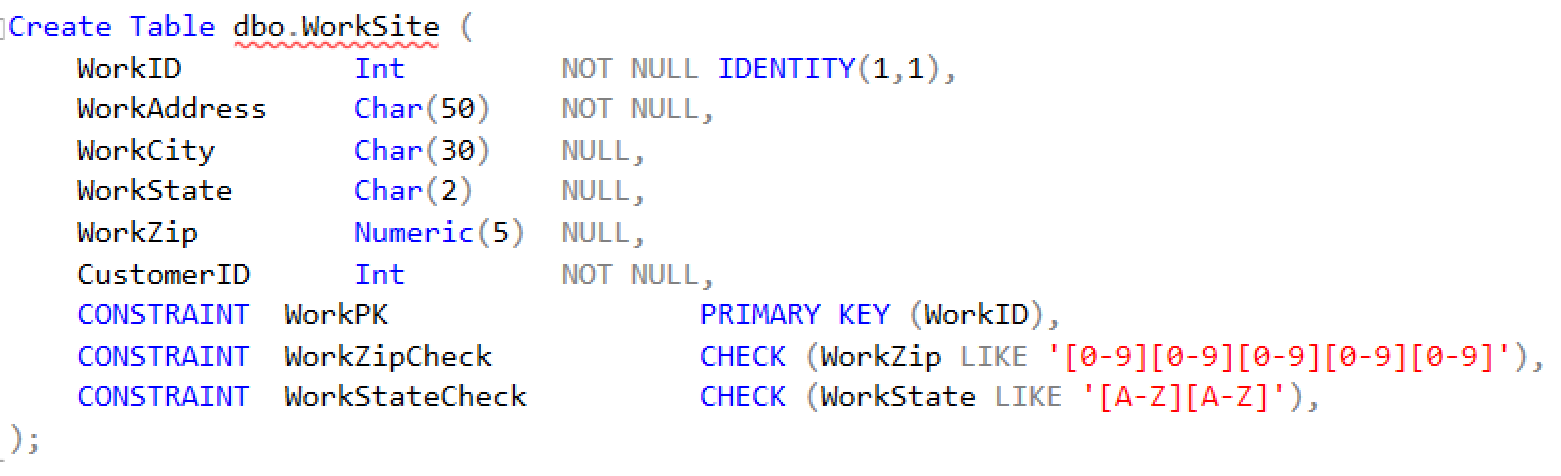


**WorkSite**

This table contains the information from each work site. One customer may have multiple quotes at multiple work sites (such as the two quotes presently stored for Allegheny City Realtors), so it’s necessary to separate the two.

The constraints used to validate the data in this table are:

* Zip must be the US standard 5 numeric digits
* State must be two capital letters

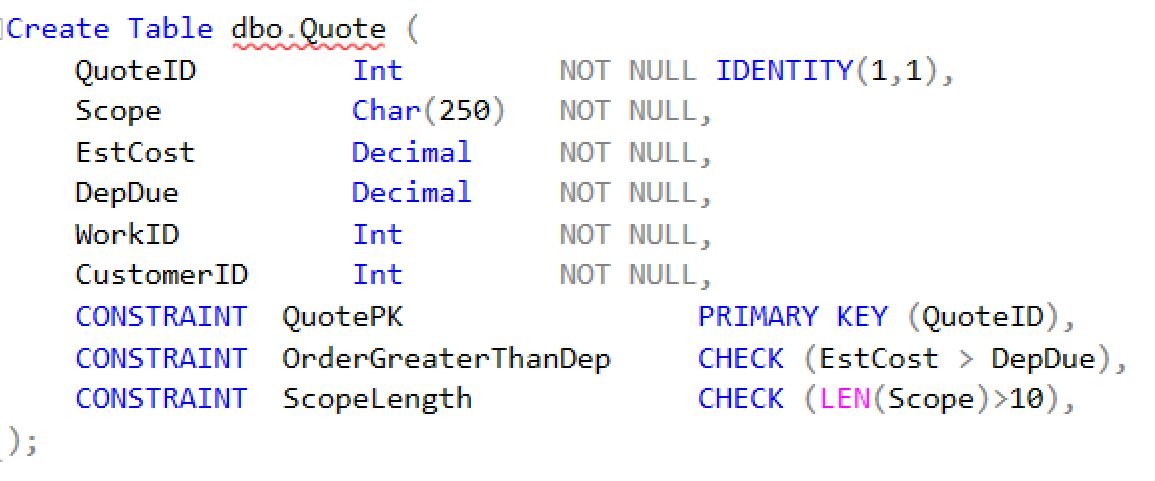


**Quote**

This table contains all information specific to each quote, including a statement of how much work needs to be done (scope), an estimate of how much it will cost, and how much of that cost is due up front upon the agreement of the quote (deposit). Because there may be multiple quotes for one work site, it’s necessary to separate the two.

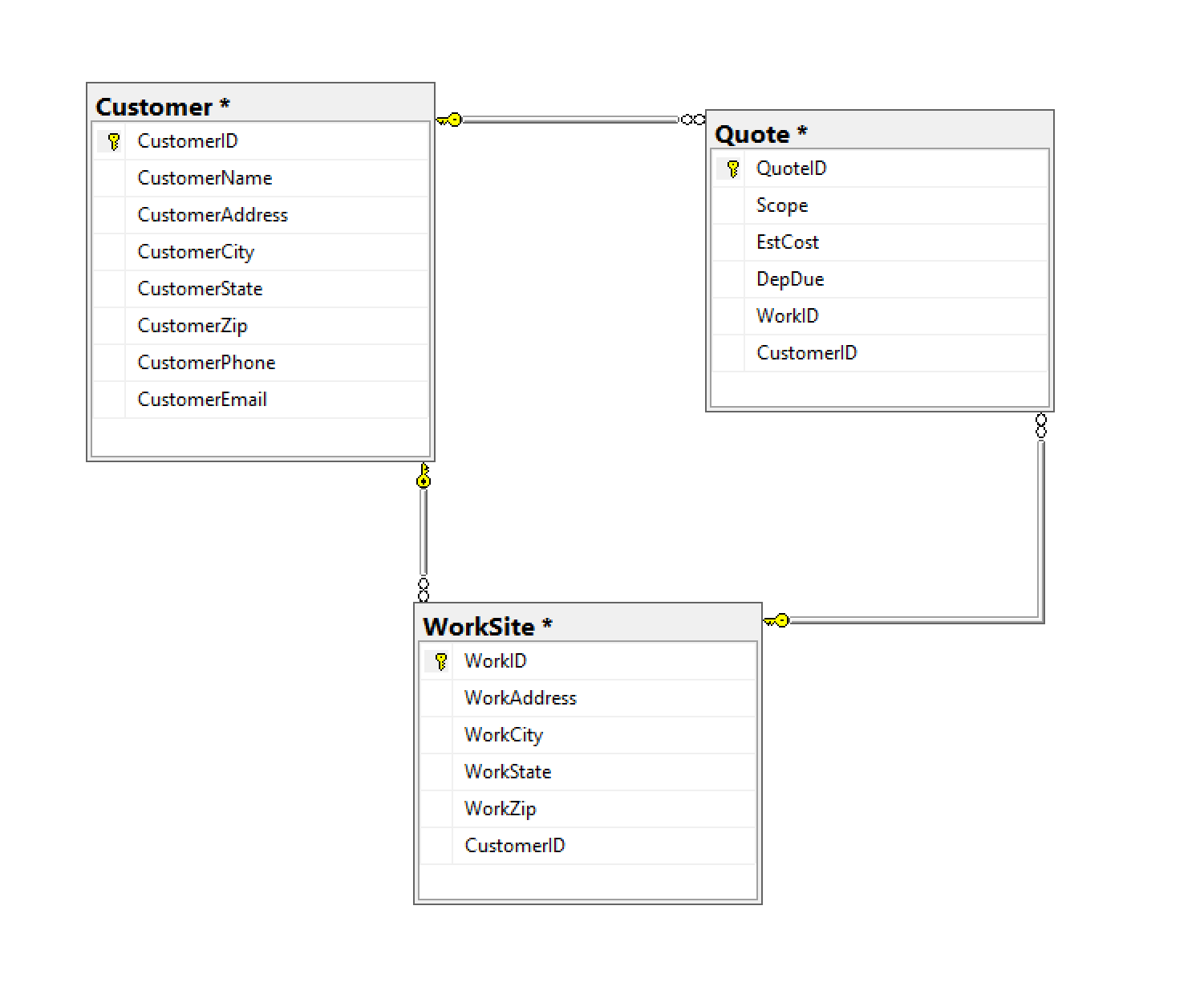
The constraints used to validate the data in this table are:

* To ensure data was entered correctly, the estimated cost must be greater than the deposit due
* To help ensure that the scope field contains meaningful information, the length must be longer than 10 characters.



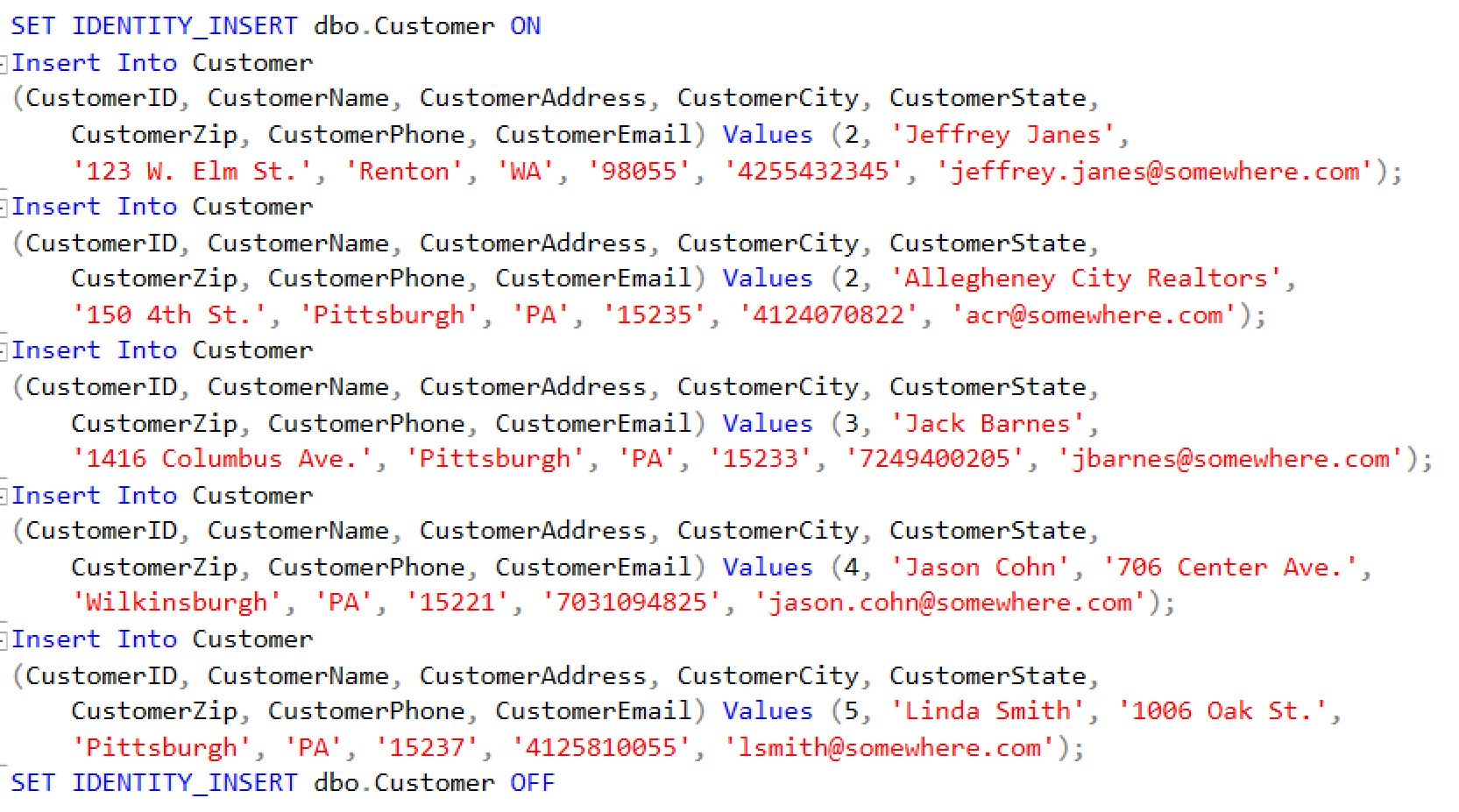
**A LISTING OF THE DATABASE DIAGRAM**

This diagram visually displays the relationships in the EWDB database.

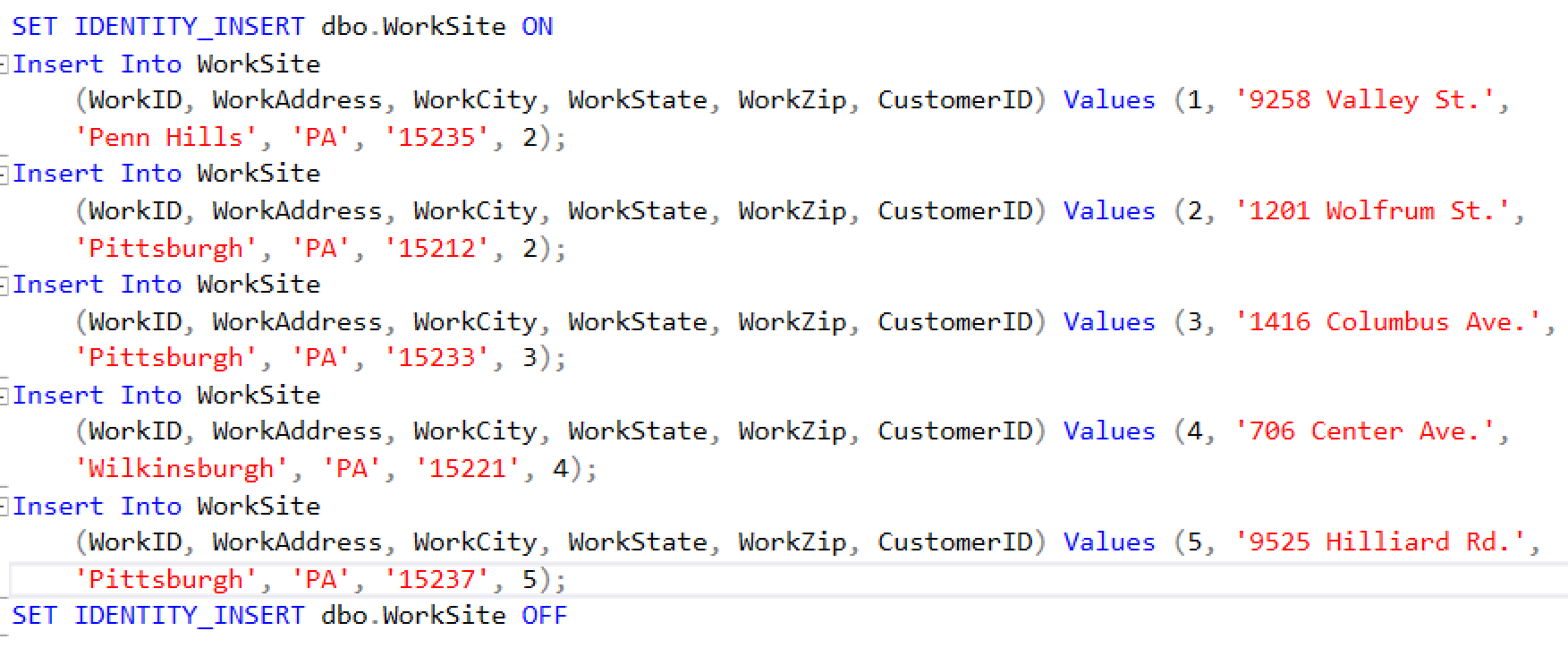


**A LISTING OF ALL OF THE DATA CONTAINED IN EACH TABLE**

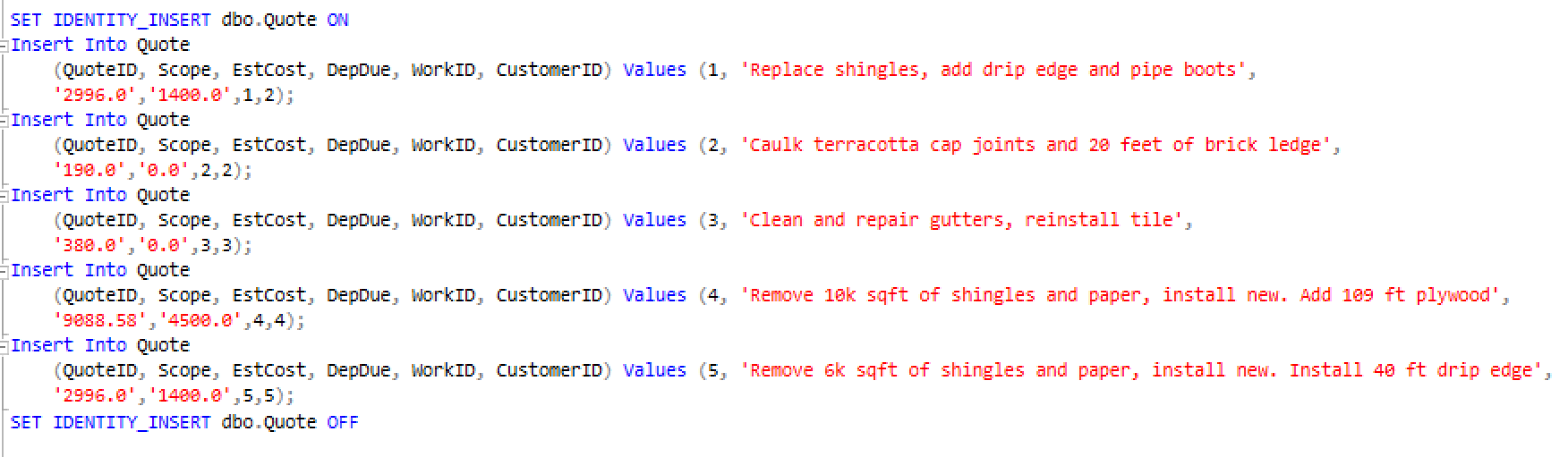
**Customer**



**WorkSite**



**Quote**



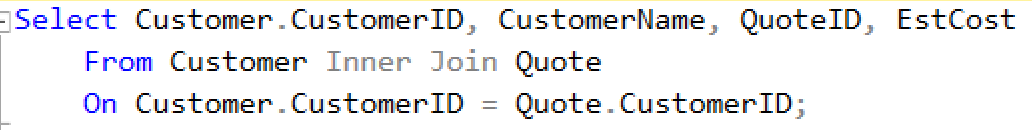
**LISTING OF THREE SQL SELECT STATEMENTS FROM TWO OR MORE OF THE TABLES CONTAINED IN YOUR DATABASE**

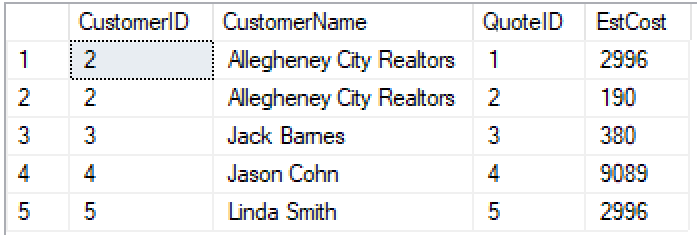
For all select statements, the following are included:

* The name of the .sql file containing the query
* A brief description of the purpose of the query
* A picture of the code itself
* A picture of the query result

**Select statement #1: AllQuotesForCustomer.sql**

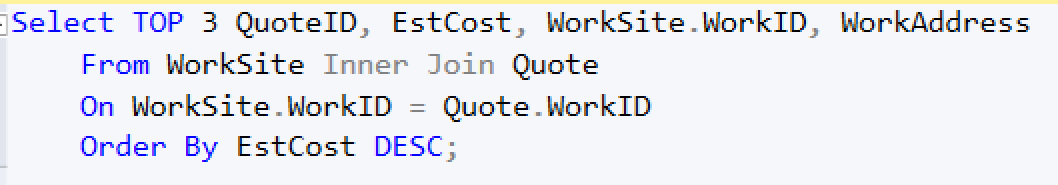
This query joins the Customer and Quote tables in order to display all quotes (plus costs) for each customer.

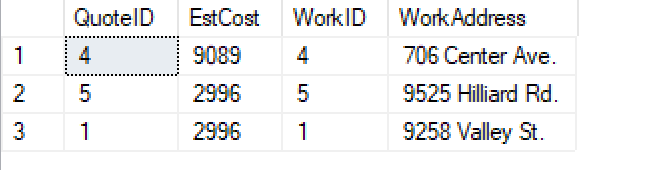




**Select statement #2: Top3HighestDollarQuotes.sql**

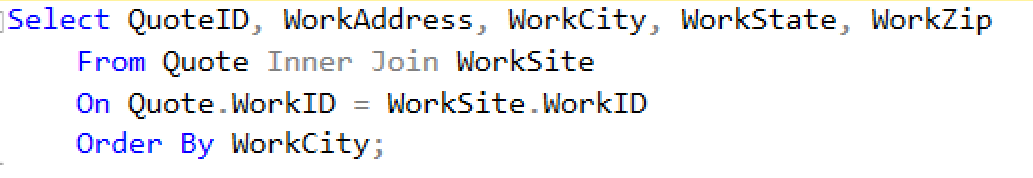
This query displays the top 3 highest dollar quotes by joining the WorkSite and Quote tables to display relevant quote and work site information.

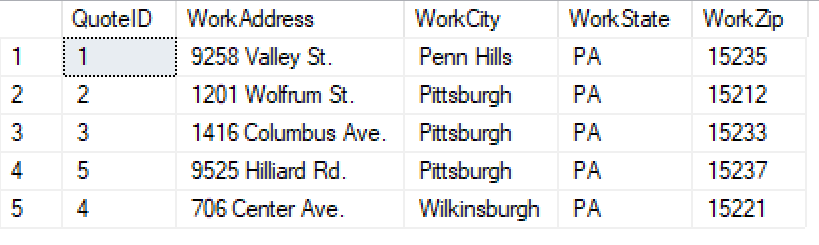




**Select statement #3: ShowQuotesByAddress.sql**

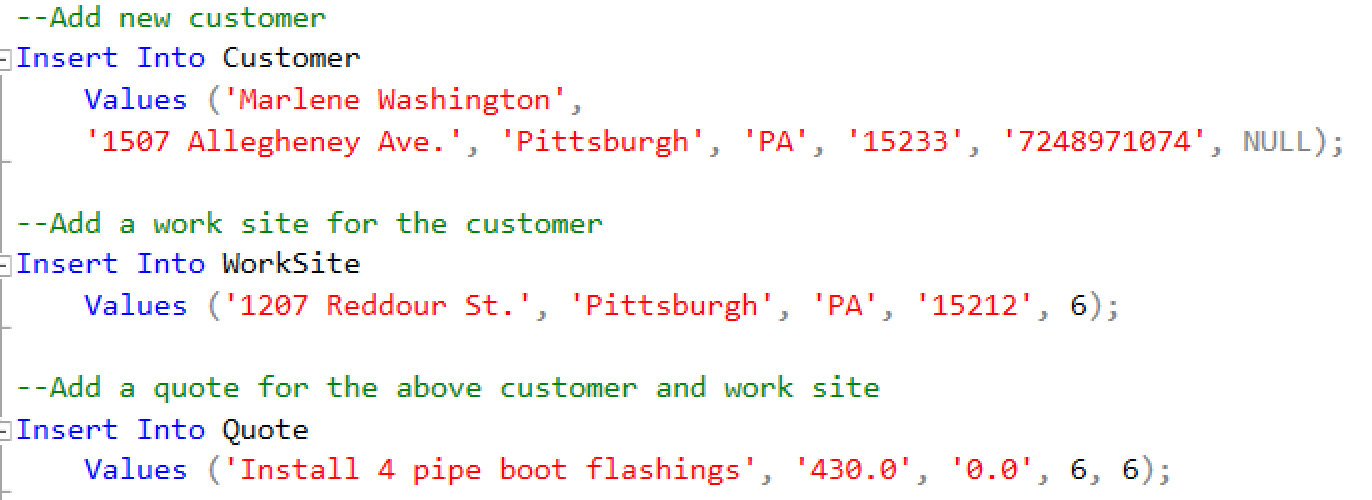
This final query would be used by employees in planning their route. It joins the Quote and WorkSite tables in order to show which quotes are attached to which addresses, then orders the list by city.



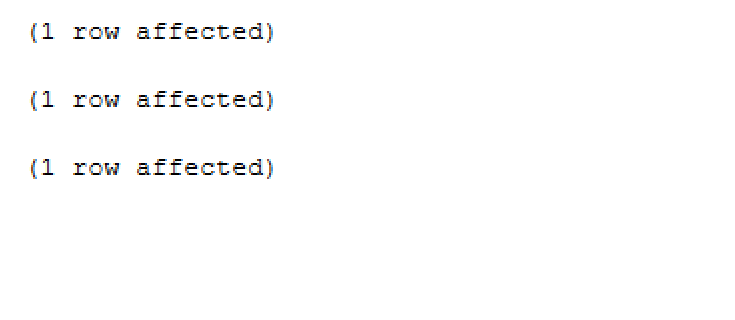


**LISTING OF AN SQL INSERT STATEMENT TO ADD AT LEAST ONE RECORD INTO ONE (OR MORE) OF THE TABLE(S) CONTAINED IN THE DATABASE**

Insert statements:



Output:



**LISTING OF AN SQL UPDATE STATEMENT TO UPDATE AT LEAST ONE RECORD IN ONE (OR MORE) OF THE TABLE(S) CONTAINED IN THE DATABASE**

Sometimes users of an application enter data incorrectly, and it’s important to be able to change it. To simulate this, I entered an incorrect duplicate value, such that quotes 1 and 5 had the same cost and deposit information, as well as similar scopes. This update statement changes the values of scope, cost, and deposit to the correct ones for quote 5.

Update statement:



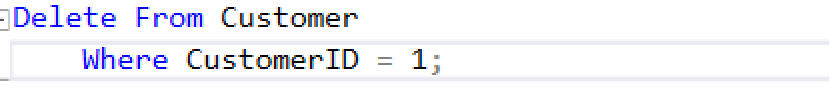
Result:



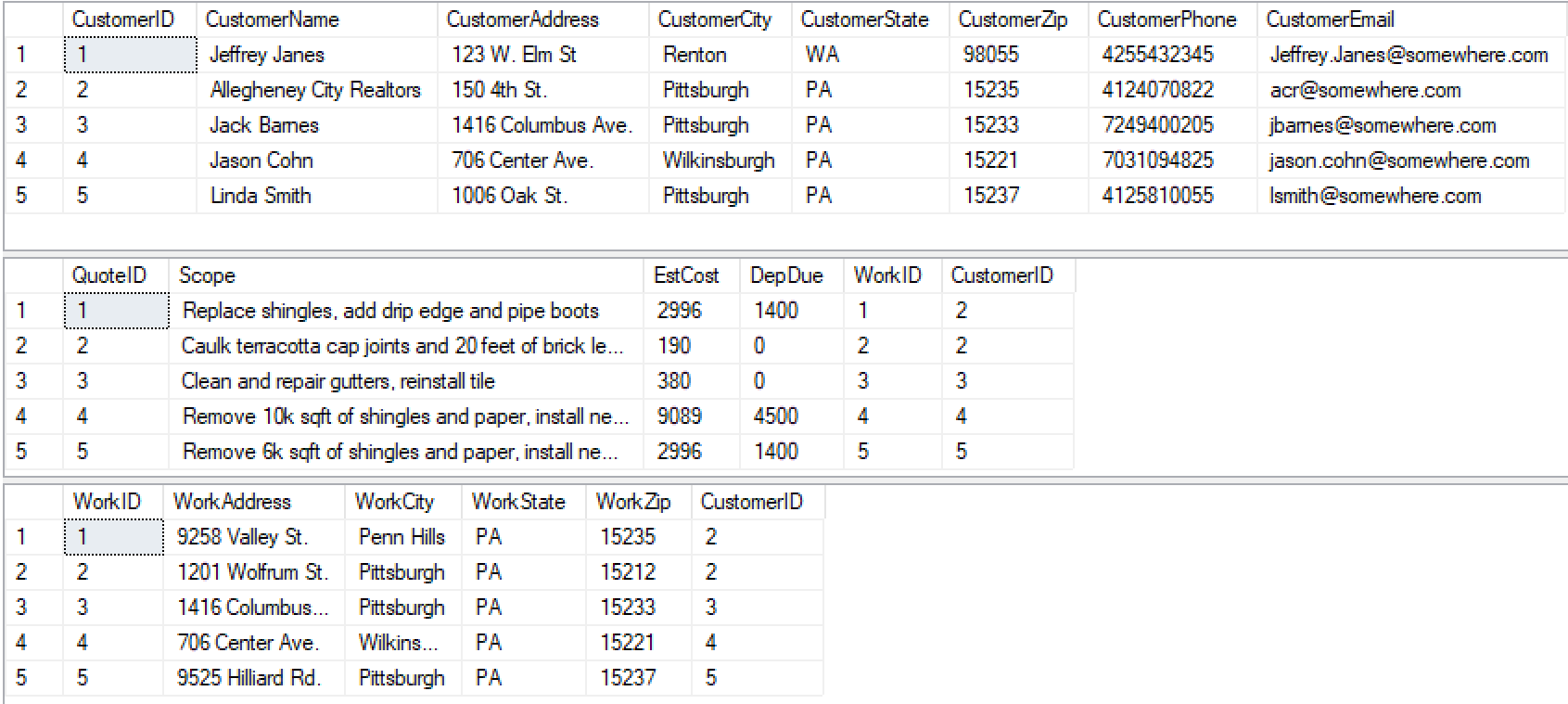
**LISTING OF AN SQL DELETE STATEMENT TO DELETE AT LEAST ONE RECORD IN ONE (OR MORE) TABLE(S) CONTAINED IN THE DATABASE**

When I populated these tables, I created a stored customer who has no current quotes (CustomerID 1). This delete statement deletes that non-current customer from the database.

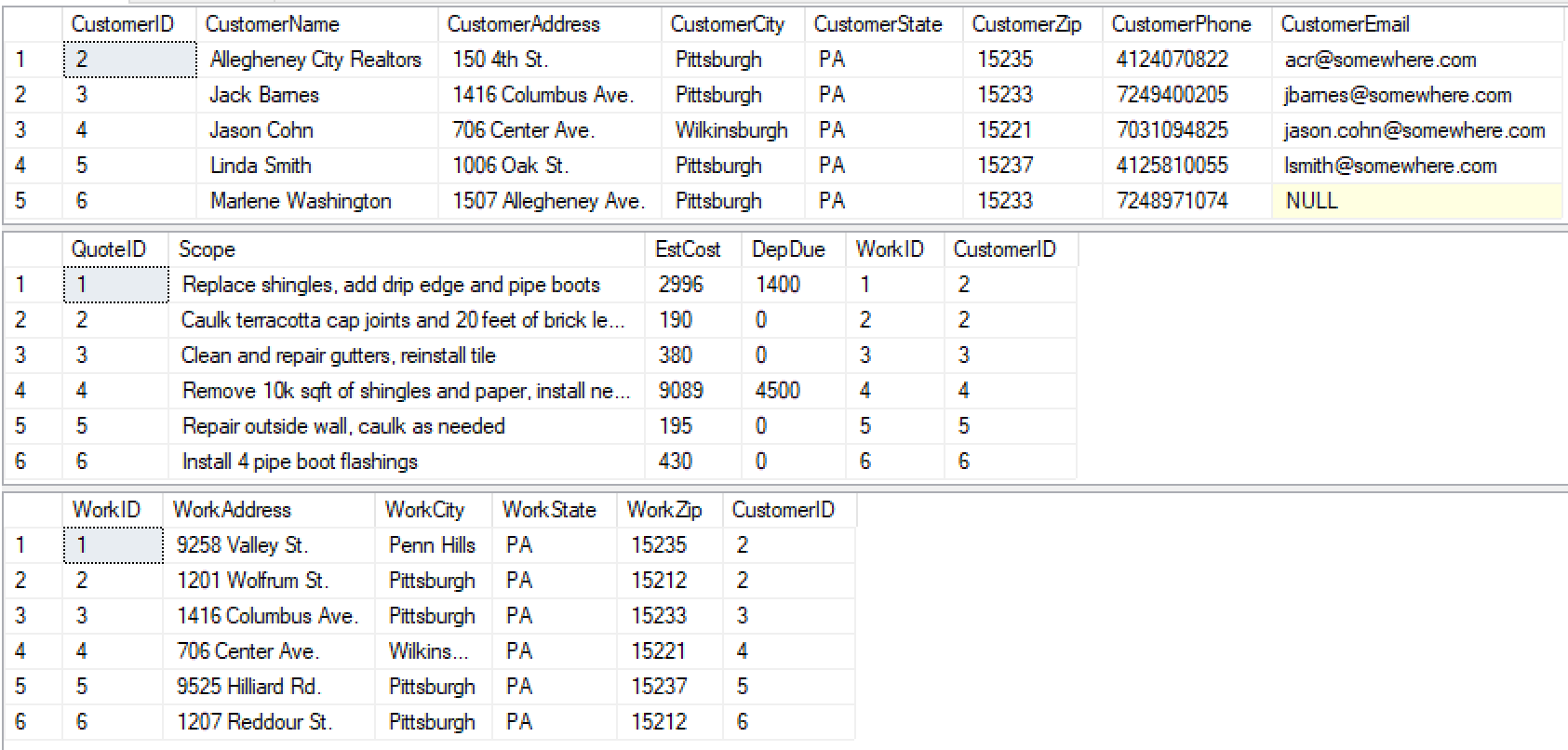
Delete statement:



For the past three sections, I’ve been modifying the data in the database, so I figured it would be useful to add a reference. Here is all the data before the modifications:



And here is the data after the modifications (next page):



**SUMMARY OF YOUR CREATED DATABASE MANAGEMENT SYSTEM**

The purpose of the Ellis Wyatt database is to track customers, work sites, and quotes. The three tables, related variously through primary and foreign keys, allow us to keep track of each of these things and display combined data using queries and select statements.

From a business standpoint, the ability to see which jobs are of the highest value through the Top3HighestDollarQuotes query is useful so that efforts can be prioritized. The ShowQuotesByAddress query is useful for planning a route from job to job, and knowing which jobs are near each other. And the ability to see relationships between quotes and customers, as in the AllQuotesForCustomer query, is especially useful for repeat business.

I was building this database not just as a class project but also as part of what will hopefully become a real-world application. For a Java class, I built a front-end application for this business, and here, I built a database structure. My next challenge will be to connect these two, so that my Java front-end will be able to store and retrieve data from my SQL database.

I found learning SQL and database structures very interesting, and I’m even more exited about the prospect of combining this with my previous programming endeavors to create a working application.