

**By**

**Sunil Raj Thota  
ALY6030 - Data Warehousing and SQL**

**Class Name: Fall 2021 CPS Quarter**

**Class Number: 202215**

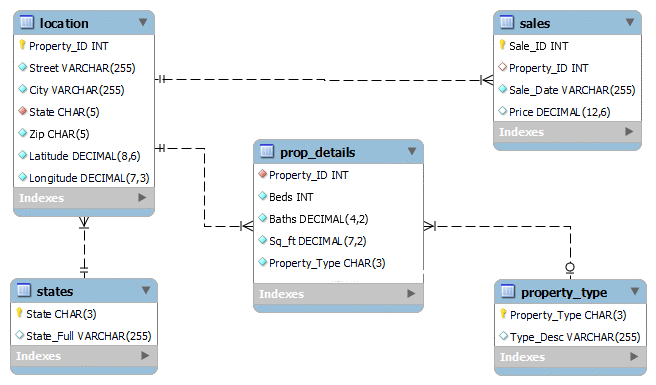
**CRN: 70526**

**Week 2 Assignment 2**

**thota.su@northeastern.edu**

**Introduction:**

In this Assignment, The report is divided into two sections. The first section covers Database Normalization. I have also shared each step with dedicated screenshots and comments where ever required. To comprehend the ideas of joins, complete this project by studying normalization stages in a database, establishing a database with at least 5 tables that satisfy at least 3NF, and developing SQL statements to address at least 3 business inquiries. I aim to perceive these ideas by using various sorts of joins, aggregating methods, and variables. For this report, I have SQL as the language to interprets the results by creating a database schema and defining the table attributes with their data types. In the given list of datasets, I have chosen the “Sacramento” data. I have also gone beyond and analyzed 5 tables and executed the SQL Query Design. This dataset has 985 records with 12 variables. Here is the ER Diagram that I have created



One aspect I was taken into consideration was whether the primary file contains a collection of sale facts, and this same property might be resold multiple times over the period. As a result, each purchase has a one-to-one connection with the location, while each location has one or many relationships with sales. Similarly, location has a one-to-one connection with the state, but because a state can have several attributes, a state has a one-to-many relationship with location. Moreover, every Property belongs to a specific property type, and each property type can have 1 or even more properties.

The location table contains information on the location, such as the street name, city, state, and zip code. The primary key would be Property ID. Property Details table contains data about the property's construction, like how many bedrooms and bathrooms it has. How big is the whole thing? Property ID from the Location table would be a foreign key connection in this table. Because there are times when one of the 2.5 bathrooms does not have a shower, I utilized the decimals datatype to allow the property to keep the.5 values if necessary.

The sales table maintains data on sales. The Sale ID is the primary key for each sale, whereas the Property ID is the foreign key. This table will keep track of which properties were sold when and for what price. Property Type table has property Types like "Residential" or "Multi-Family" takes a long time, this table will record Property Type acronyms in three characters. This database was built to contain state abbreviations. We presently only have details for one state, but as the database grows, this will become more helpful. The primary key will be the State characteristic, which stores the acronym, and the foreign key will be the Location ID.

**Analysis:**

The Analysis part is divided into 2 sections;

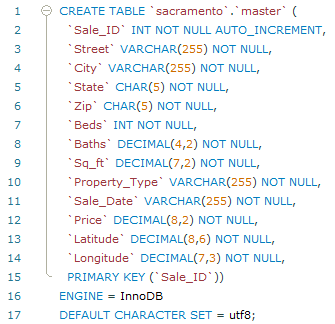
**Part 1: Database Normalization**

For the analysis part, I have chosen the “Sacramento” dataset

Step 1: After opening the MySQL Workbench and setting up the Server connection by giving a password. We will be seeing an IDE kind of workspace to do our analysis using SQL. Creating a database schema called “sacramento”

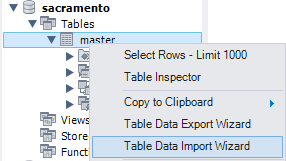


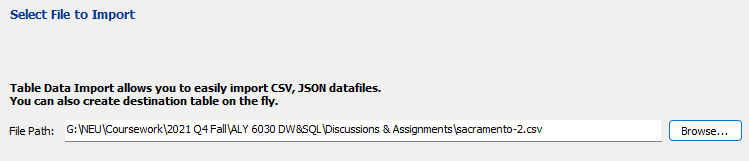
Step 2: Let’s create a table to store the Sacramento Data by defining their datatypes and the size of each attribute.

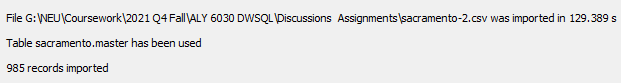


Step3:

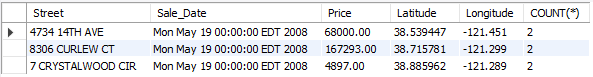
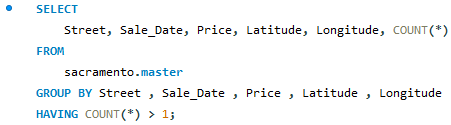
The next, step is to import the CSV dataset by using the “Table Data Import Wizard” option as shown below.





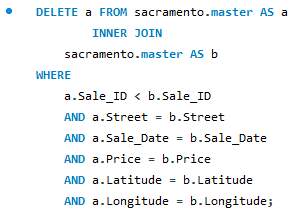


Step 4: Now, I will check for the duplicates and ensure there is no data redundancy.



From the above results, we can see that there are 3 duplicate records in the dataset.

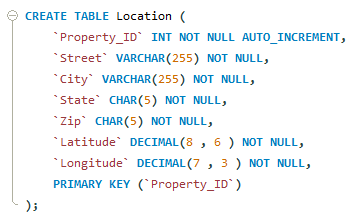
Step 5: Let’s delete the duplicates by performing an INNER JOIN and store only uniques records.



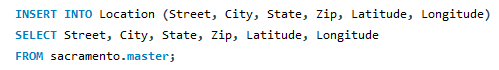
After running this code, I have observed that the total number of records is 982 now.

**Part 2: SQL Query Design**

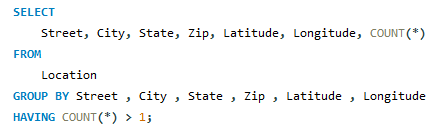
Let’s create tables with similar data types for these columns as I have performed before. For this, I will add a new field called Property\_ID as the PK.



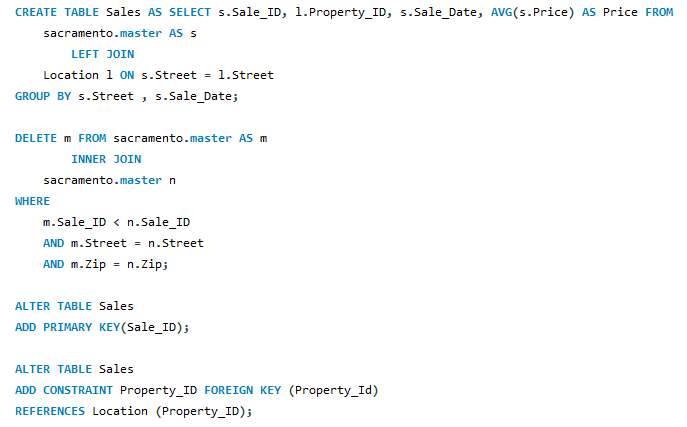


****

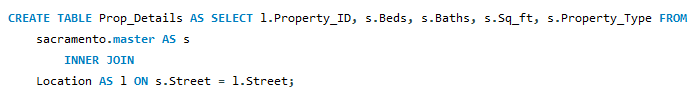
Now, let’s check whether if there are any double entries in the dataset.

****

From the results above, we can see that there is only 1 record with 2 sales entries and deleted the extra one.

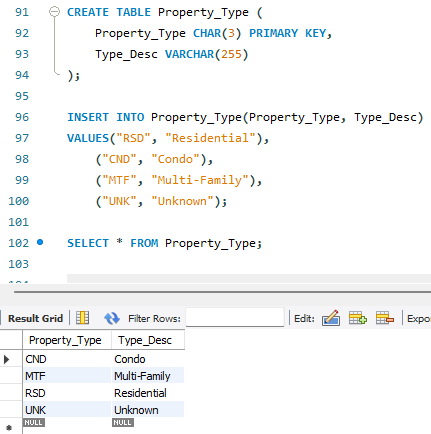
****

From the above screenshot, we can observe that one single case of the same property being sold two times on the same day is a $65 variation, then I used the group by on Street and Sale date only used the AVG of the cost. This will upgrade both data with the new average price, and one of them will need to be removed in the next stages.

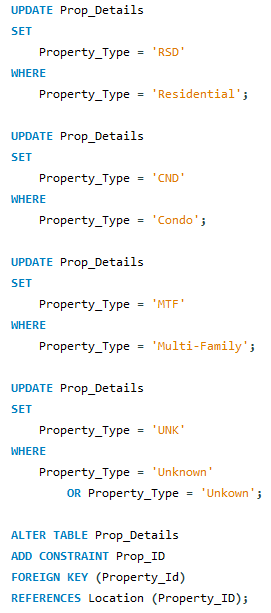
****

From the above pic, To record the Property ID, beds, baths, sq ft, and Property Type, I'll build a different table by combining the temporary table, sacramento main, and Location. The sacramento main table lacks a property ID column, but we need to know which property the property dimension refers to, so I used an inner join to build the relationship.

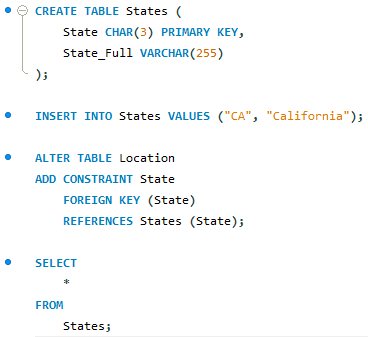
Now, let’s store a variety of types of Property in the Property\_Type table



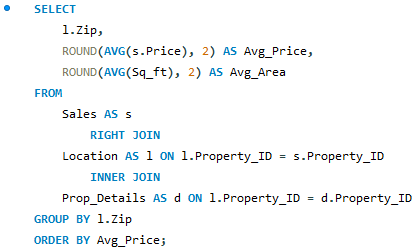
Now I will update the Prop\_Details table to store the 3 character code and will modify the type of the data to be equivalent to the Property\_Type field and add the reference key.

****

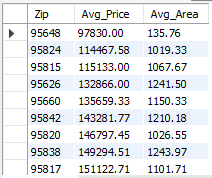
For the time being, I'll include only California in the state table. If the database expands to incorporate property details from other states when it comes, this table will be included as a perspective of development. The state property will be used as a Foreign Key in the Location database once we construct the state table and insert California.

****

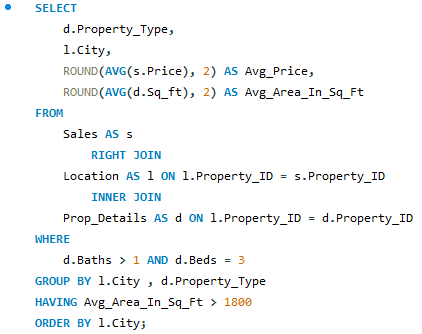
1. **Business Questions:**
2. **What are the average property prices and square feet in each zip code?**



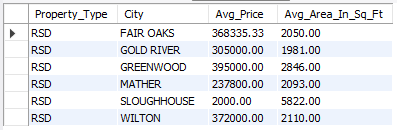
As I only want Properties with all of the property details like Sq ft, bed, and so on, I used an inner join between Location and Property details. As my focus is on zip code prices, I used a RIGHT JOIN between Sales and Location. The zip codes 95648, 95824, and 95815 have the lowest average prices.



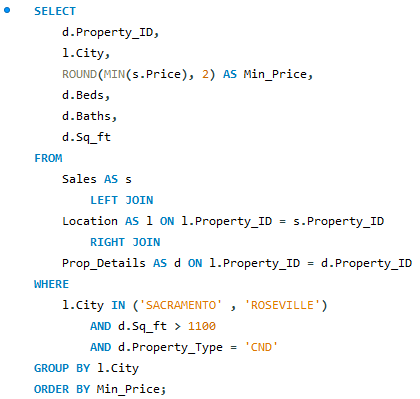
1. **In each City and Property Type, what is the average price of a home with a 3-bedroom and more than a bath that is having an average square foot of more than 1800?**



By joining sales, location, and Property details tables, the above query will populate Property Type, City, Average price, and Area. I used a circumstance in the where clause to limit the results to properties with three bedrooms and more than one bathroom, and the Having clause will only return data with an average area of more than 1800 square feet.



1. **In SACRAMENTO and ROSEVILLE, what is the MIN price of a condo over 1100-square-feet?**

****

I have used the MIN aggregate function to find properties with the lowest prices, and I added 3 elements to the where clause to include condo properties in Sacramento and Roseville with an area greater than 1100 square feet.



After all this analysis, I will DELETE the initial sacramento table.



**Conclusion:**

I am glad to choose this dataset because it allowed me to experiment with various methods for creating tables and populating records, such as joins and the INSERT INTO functions. While I am pleased with my approach, I wonder if the tables were created using a long process and if there is a more efficient way? If Excel manipulation was permitted, I would pre-add the primary key attribute in CSV and possibly create the tables more quickly because the Import wizard would work seamlessly.

Another limitation of the business questions section was that because all three tables—sales details, property details, and location had the same number of records or matched records, no matter which joins I used, the outcome was the same. I'd like to try my hand at a more complex dataset with no matching records in each table so that I can double-check my reasoning while utilizing joins.