Eric Dishmon

DSBA 6520

Sean Robinson

8/9/2021

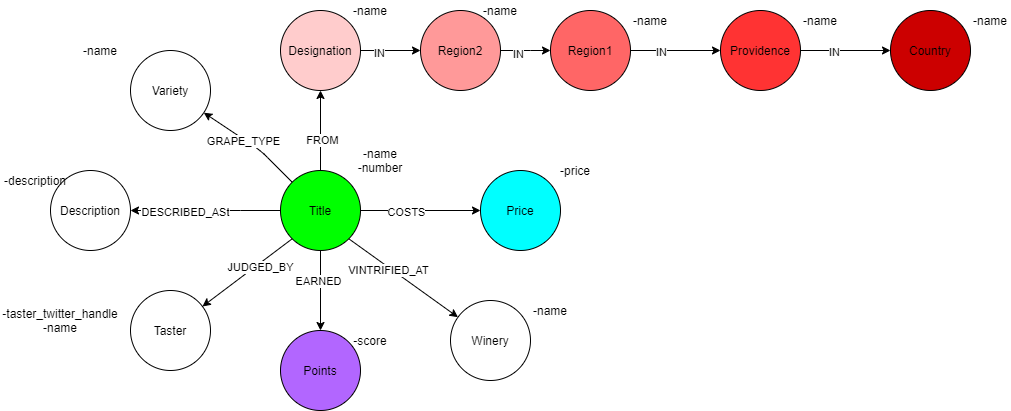
Dishes by Dishmon: A Wine Network

**Background:**

As the new owner of a restaurant, providing the best experience for my patrons is on the forefront of my mind when making decisions. Considering different factors from the ambience, to the food, to the overall price of a meal, customer satisfaction is very important. I have already created the menu for opening night, but I have not selected wine pairings to go along with the food. Luckily, I have a large wine dataset with plenty of information to help me identify the best options based on reviews and price to go with my food. Although there is more work upfront, I am going to design a graph data model so that I can more easily find the best wines to go with each food for all of my future menu items. I will also use this graph to be able to identify key regions or wineries that I will want to partner with as my restaurant grows. The end goal for all of this is to have high customer satisfaction so that more people know about my restaurant, more people will come back to my restaurant, and the business will do better overall.

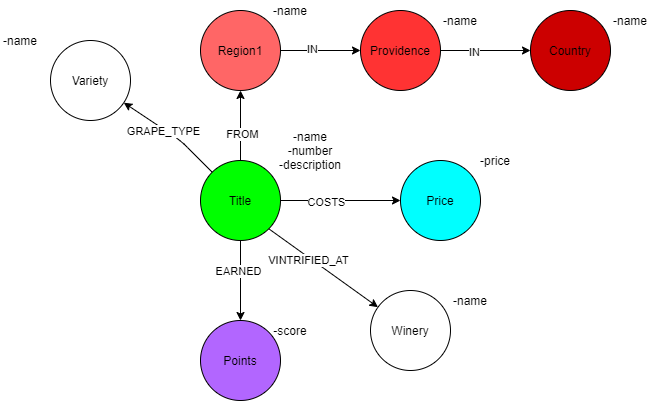
**Data Model and Database Setup:**

I decided that creating a model of the dataset before trying to import it into Neo4J would be the best course of action because it allows me to see the relationship between the variables that I want to explore. I went through several iterations of the data model throughout this project as I removed superfluous information to find the best wines for my restaurant. To begin with, I kept all of the variables within the dataset and created Model 1.



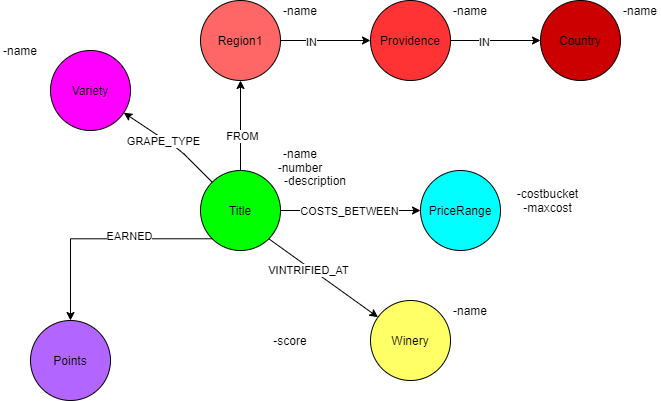
*Model 1*

I quickly realized that some variables were not needed for my purposes, so I revised Model 1 so that it was more compact and contained only the most important variables. Model 2 reflects these changes that I made after the database setup stage. Some of the changes made were removing the Taster variable completely, as I am only focused on the actual reviews of the wines, and moving the Description node to be a property of the Title node. Finally, the Designation and Region2 variables were removed due to there being too many missing values in the dataset.

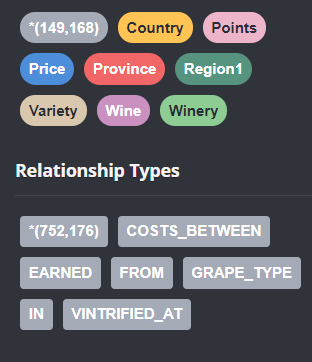


*Model 2*

My final iteration to the data model came after creating my initial queries in Neo4J. I realized that there were too many Price nodes to have any meaningful insights for the model, so I decided to create ranges for the prices to put them into buckets. I chose increments of $10 as the bucket size up to $140, and beyond $140 was a single bucket. The bucket size was based on the very skewed data towards under $140. I also added a property to the Price node called maxcost. This made it possible to order the results based on a numeric value while still being able to create string based nodes. After making these changes, Model 3 was the final version of my graph model used for all analyses and created the following model and database in Neo4J.



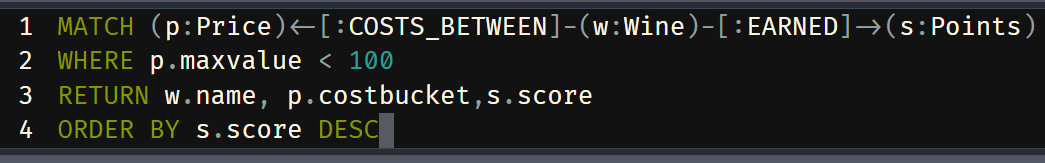
*Model 3*

****

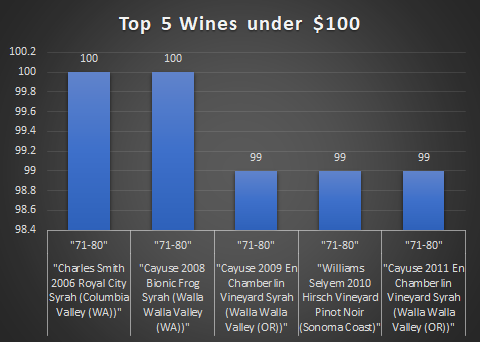
*Model 3 Neo4J Database*

**Queries:**

After completing the setup of the database in Neo4J, I wanted to gather some basic information about some topics that I was interested in for my restaurant. First, I wanted to see what the top scoring wines were for under $100 a bottle. I never want to purchase a bottle of wine over this amount because it would raise the price of the meal too high and I want my customers to feel comfortable coming to my restaurant for any occasion, not just special ones. The code and top five results for my query is as follows:

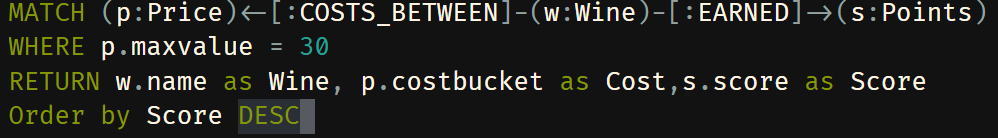


*Query 1 Code*

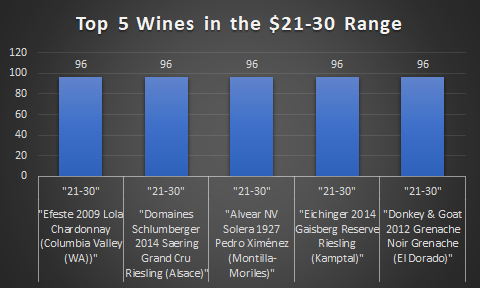


*Query 1 Results*

From my Query 1, I found out that there are only two wines that scored 100 points that are under $100, but more importantly, I found out that top scoring wines do not have to cost an egregious amount of money. This led me to create Query 2, one that identifies the top scoring wines in the median price range of $21-30. The code and results for Query 2 are as follows:

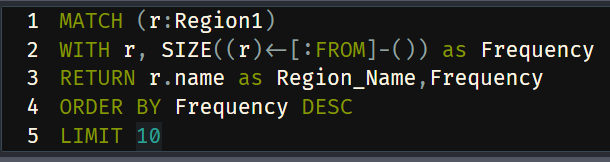


*Query 2 Code*

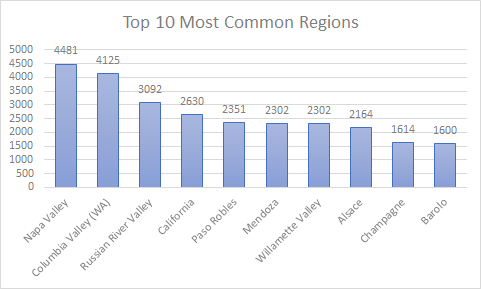


*Query 2 Results*

Query 2 showed that even for a lower price there are still high quality wines available. I would try to choose a cheaper wine with a high rating compared to a higher priced wine of similar rating so that I can pass those savings along to my customers. Finally, just to begin researching where the wines came from, I created Query 3. Query 3 shows me the top ten most common regions for wines produced in the dataset. Given that all of these wines scored 80 or above, I can focus my attention on the larger markets to save time on travel without sacrificing quality. The code and results for Query 3 are as follows:



*Query 3 Code*

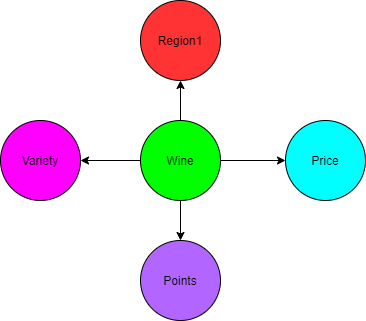


*Query 3 Results*

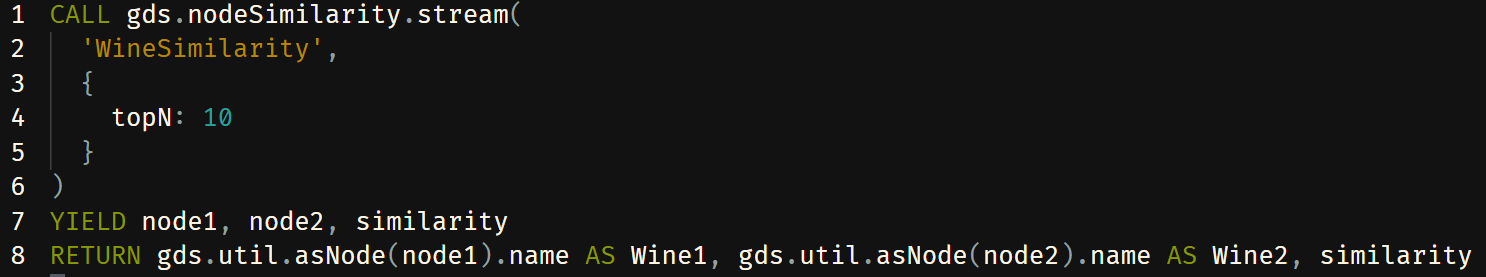
Based on the results, I will focus more on California and the Pacific Northwest as regions to consider for wineries so I do not have to travel internationally or pay for high shipping costs. After creating my three queries, I decided to move on to algorithms to try to establish groups within the data to more easily segment the wines.

**Algorithms:**

I wanted to establish some ways of comparing wines to each other based on some key categories: location, price, score, and variety. The thought process behind this was if I found a wine that I liked, I could easily find other wines that were similar to it. My first two algorithms dealt with this problem directly by using the node similarity algorithm to give a numeric value to the similarity between two wines. Algorithm 1 and Algorithm 2 are made from the same projection listed below. The difference between the first and second algorithm is that Algorithm 1 compares each of the wines to every other wine in the entire dataset and shows how similar they are on a scale from 0-1. Only the top 10 results are being shown due to processing power limitations. Comparatively, Algorithm 2 shows only the most similar wine for each wine, greatly reducing the volume returned. I wanted to use both of these algorithms because I think it is important to be able to fully compare all of the wines based on similarity, but it is also more convenient to have the most similar wine readily available. The projection, code, and results for the two algorithms are as follows:



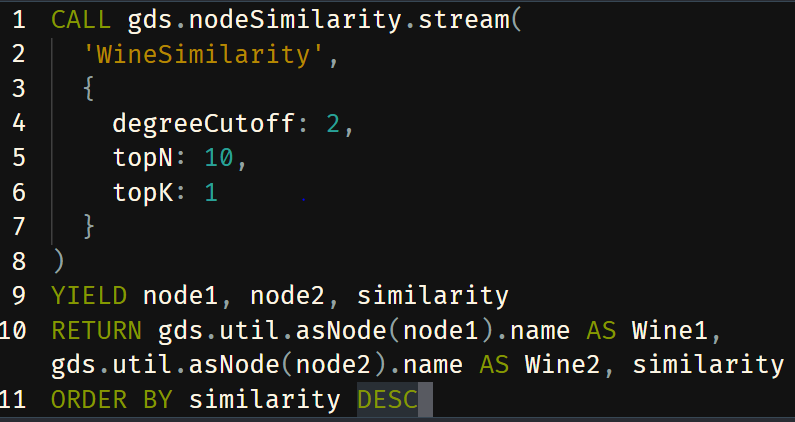
*Projection for Algorithms 1 and 2*

****

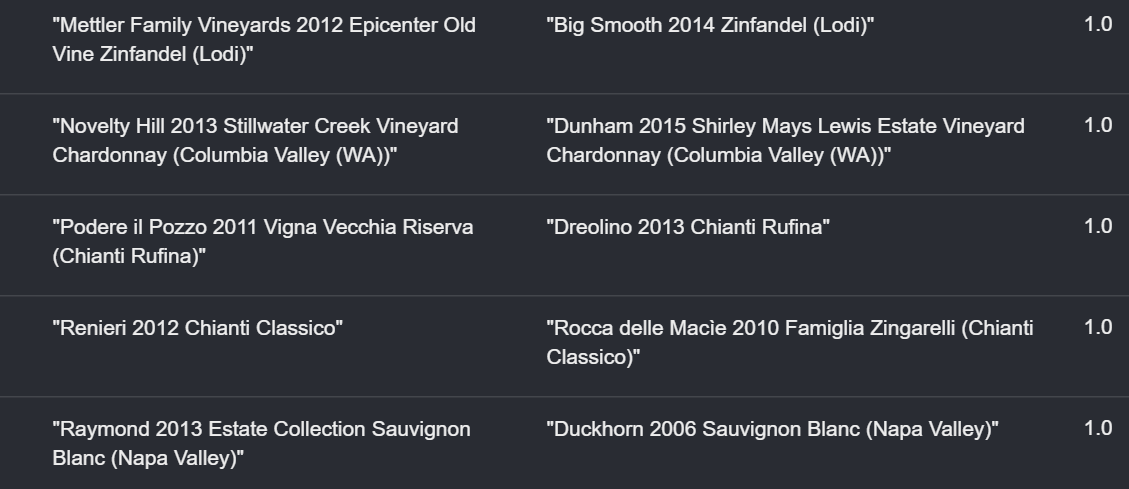
*Code for Algorithm 1*



*Results for Algorithm 1*

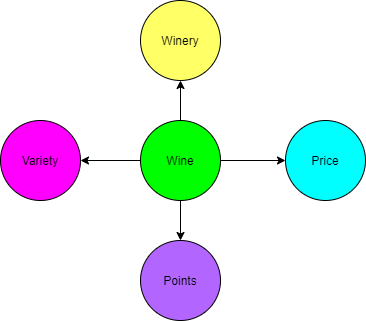


*Code for Algorithm 2*

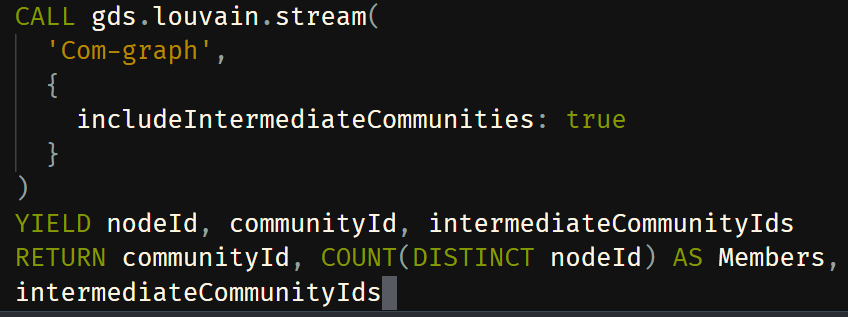


*Results for Algorithm 2*

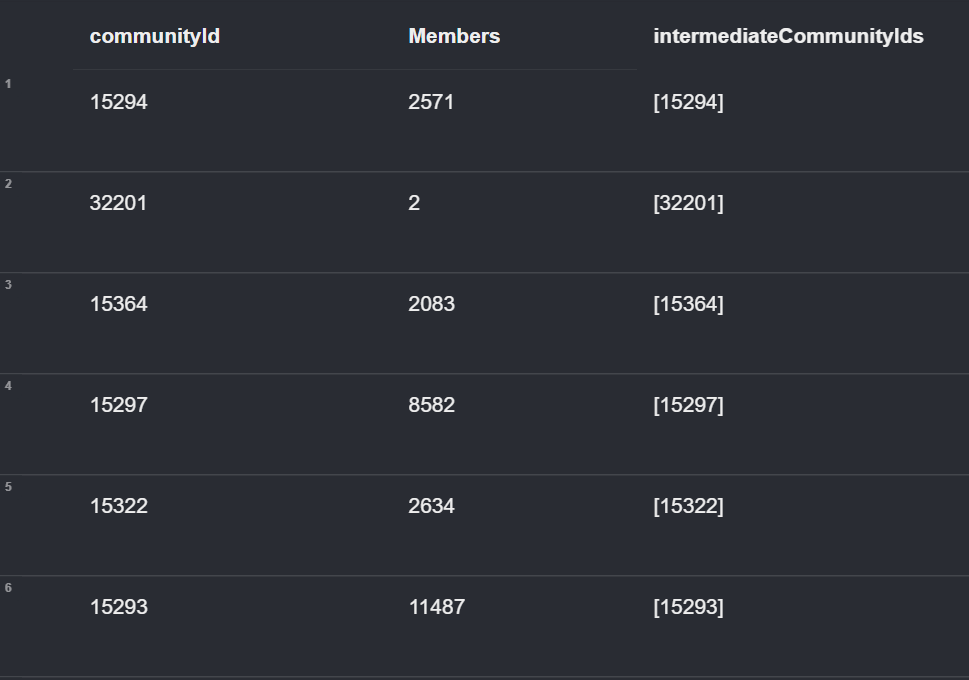
After seeing these results, I took another approach to finding wines that are alike. I decided to use a Community Detection Algorithm, Louvain, to create communities within my graph database. This differs from the previous algorithms in a few key ways. One, I used a different projection that instead focuses on the specific winery, not just the region. I did this so that wine from the same wineries could be grouped together. Two, this does not directly compare each wine to each other, but rather groups wines together into similar communities. These communities are smaller than a comparison to each and every wine, but larger than just returning the top match. I will use this when I want to create a list of wines to choose from based on a wine that I already like. The projection, code, and results from Algorithm 3 are as follows:



*Projection for Algorithm 3*



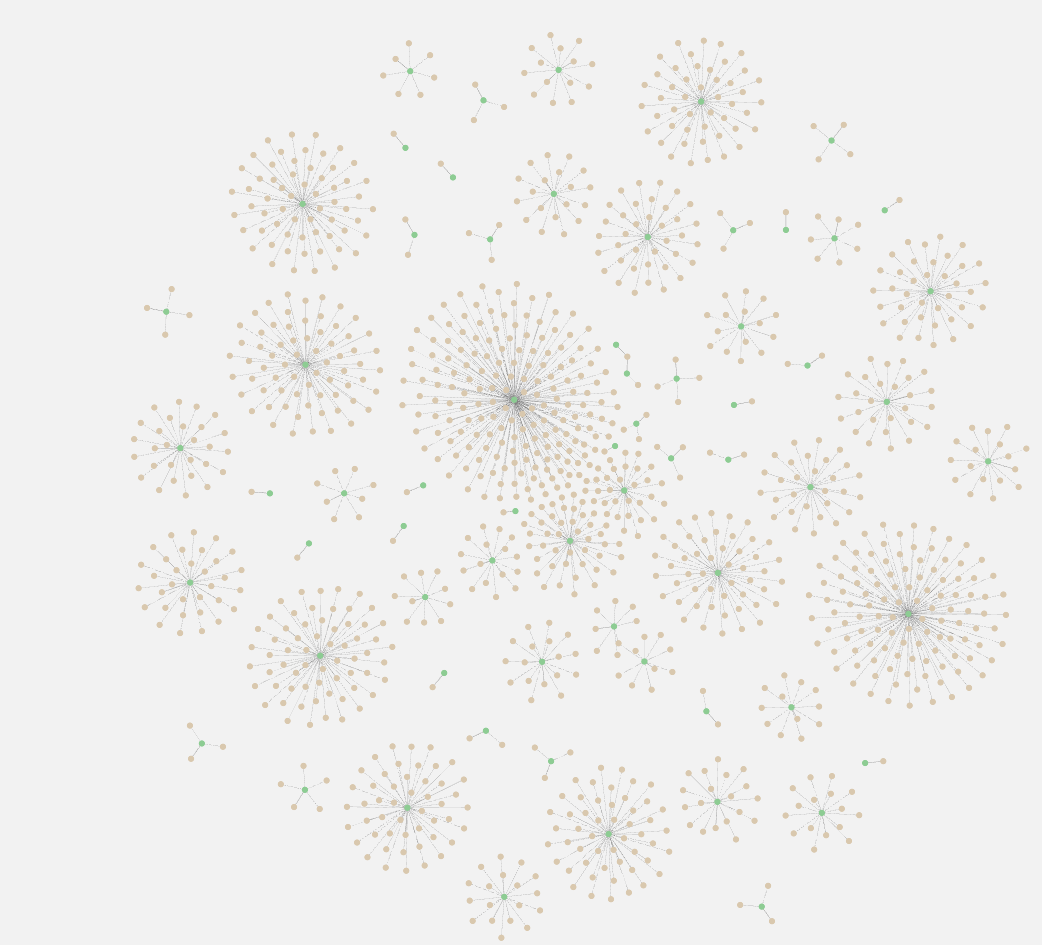
*Code for Algorithm 3*



*Results for Algorithm 3*

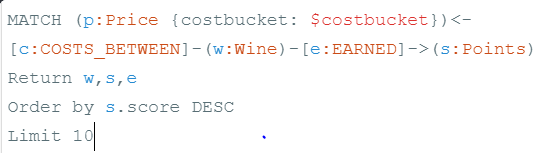
**Visualizations:**

After utilizing the queries and algorithms to create lookup tables, I wanted to create some simple visualizations and search phrases to be able to quickly use the graph database in the future. This will make it a lot easier to find out information in the database compared to having to create more queries and algorithms in the future. I first created two visualizations to be able to reference back to in the future. The first visualization shows different Region nodes clustered together by Province. This is important because it will make it easier for me to identify where most of the Regions are located by a grander scope so that I can focus on the areas of highest density. Although zoomed out currently, I would be able to see the sames off all the Province nodes in the software.



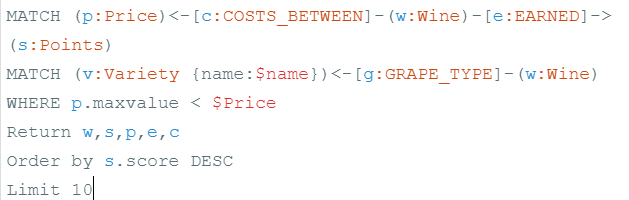
*Visualization 1: Regions by Provinces*

My second visualization took the opposite approach. I wanted to focus on the highest rated wines at the lowest prices. As shown above in Query 2, I discovered that the cheapest wines with the highest scores were in the $71-80 range and both of the Syrah variety. In the visualization it shows the specific wineries in which they were vinified, so that I can easily contact those wineries. I would save these for a special occasion, like the grand opening of my restaurant. After creating these visualizations, I wanted to create search phrases so that anyone using Neo4J could easily look up information with minimal guidance. I created the first search phrase, “Best Wines in $costbucket Price Range” which returns the top ten wines based on score in a certain price range. This is super useful to be able to easily find the best wines in a certain price range. If I know about how much I am willing to spend, this is a great option to quickly find the best wines in that range. The code for this search phrase is as follows:



*Search Phrase 1 Code*

My second search phrase is also based on price and points, but adds another element to the mix, variety. Search Phrase 2, “Find the Best Wines in the $name Variety under $Price Dollars”, finds the best wines of a certain variety under a price point. I will use this when I am trying to find a specific type of wine to pair with a dish that I want to keep under a certain price point. For example, if I know I am cooking a seared salmon dish, I would choose a Chardonnay to go along with it. I know that the glass of wine and the dish will sell for $40 and the price of the ingredients are $12. I would choose a bottle for $30 or less so that I can still make a profit on the dish. I see myself using this search phrase most often when trying to create an event menu with set prices and costs. The code for Search Phrase 2 is as follows:



*Search Phrase 2 Code*

**Conclusion:**

After taking the time and effort to construct a graph database, I was able to identify and create a sustainable way to pick the perfect wine for every occasion. I now know different points to expect for different prices, which wines are the best for the lowest prices, and how to find wines that are similar to a wine that I like. I also have an easy way to find wines for any occasion and know that I am getting a good deal, whether it be after a menu is created or just as pairings in general. Finally, I can begin to explore different places where I can start to establish relationships with winery owners for a sustainable supply for my restaurant. All of this will save time and money while allowing me to create a better experience for all of my patrons so that they will continue to enjoy my restaurant. As for my menu for opening night, I have decided on the following:

Appetizer: A goat cheese charcuterie board and meat platter. Paired with Barnard Giffin 2012 Fumé Blanc Sauvignon Blanc

Main Course- Osso Bucco served roasted vegetables. Paired with Charles Smith 2006 Royal City Syrah

Dessert: Chocolate Lava Cake. Paired with Taylor Fladgate NV 325 Anniversary Port