|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dublin&#39;s €10m Liberties distillery opens | Scotch Whisky | | | | |
|  |  |  | |  |
| Iowa Whiskey Sales  PRoject Deliverable II  GitHub: https://github.com/StevenEJordan/Network\_Science | | |
| Steven Jordan | | sjorda41@uncc.edu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Empty Bourbon Barrel | Kings County Distillery | | | | |
|  |  |  | |  | |
|  | Business Use Case  I am putting myself in the shoes of a Data Scientist working for a startup whiskey distillery tasked with understanding and optimizing the retail sales channels. Specifically, we would like to begin distribution in Iowa but don’t know what vendor to use and which stores/areas to target with our initial launch. Our product is American made but in the Irish Whiskey tradition. This means we need a distributor who specializes in that type of product. Currently, all liquor sales go through the state who work as an intermediary between vendors and liquor stores. This means that the department can help us understand what items are selling where and what vendors are supplying them. This data will allow us to answer our business questions around distribution and marketing. | |  | |

|  |  |
| --- | --- |
|  |  |
| iowa sales DataData Source Our data is hosted on BigQuery in their public datasets, in the “bigquery-public-data.iowa\_liquor\_sales.sales” table. This table is updated monthly and has over 21.6 million records each of which is an individual product purchase. It has 24 columns that fall into the below three categories:   * Location of stores purchasing the alcohol * Information about the product * Metrics around quantity and price.   This dataset has many fields and records that aren’t useful for our analysis so we will need to filter and reform it a bit. Exporting the Data - SQL Query Since this is currently stored in a relational database, the easiest and most efficient way to transform the data is in a sql query. To accomplish this, I’ve written the below query. | |
| SELECT  upper(store\_name) store\_name,  upper(city) city,  zip\_code,  upper(county) county,  upper(category\_name) category\_name,  upper(vendor\_name) vendor\_name,  upper(item description) item\_description,  sum(pack) pack,  sum(bottle\_volume\_ml) bottle\_volume\_ml,  round(sum(state\_bottle\_cost),2) state\_bottle\_cost,  round(sum(state\_bottle\_retail),2) state\_bottle\_retail,  round(sum(bottles\_sold),2) bottles\_sold,  round(sum(sale\_dollars),2) sale\_dollars,  round(sum(volume\_sold\_liters),2) volume\_sold\_liters,  round(sum(volume\_sold\_gallons),2) volume\_sold\_gallons  FROM `bigquery-public-data.iowa\_liquor\_sales.sales`  where date >= "2021-01-01"  and upper(category\_name) like "%WHISK%"  group by 1,2,3,4,5,6,7 | |  |
| This query has limited the data to the most recent (this year) and only showing categories that are related to whiskey since this is our business. This now has ~99k records which is much more reasonable to work with. Once completed I exported the data to a csv file for it’s upload to Neo4J. | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| Graph Data ModelsInitial Data Model My first data model had a few issues that I discovered after creating the cypher query and inserting the data.   1. Storing the values on the relationships instead of nodes made querying more difficult. 2. The invoice and item number level of detail was not needed for my analysis. The extra nodes decreased the performance and made the visualizations more confusing.    New Data Model My new data model is much more agile and will allow me to focus on what specifically I’m trying to analyze. In this case, the relationship between locations, stores, and purchases.   Cypher Query //CREATE CONSTRAINT ON (n:StoreName) ASSERT n.StoreName is UNIQUE;  //CREATE CONSTRAINT ON (n:City) ASSERT n.City is UNIQUE;  //CREATE CONSTRAINT ON (n:Zip) ASSERT n.Zip is UNIQUE;  //CREATE CONSTRAINT ON (n:County) ASSERT n.County is UNIQUE;  //CREATE CONSTRAINT ON (n:CategoryName) ASSERT n.CategoryName is UNIQUE;  //CREATE CONSTRAINT ON (n:VendorName) ASSERT n.VendorName is UNIQUE;  //CREATE CONSTRAINT ON (n:ItemDescription) ASSERT n.ItemDescription is UNIQUE;  :auto using periodic commit 1000  LOAD CSV WITH HEADERS FROM 'file:///Iowa\_Whiskey\_Sales\_Clean.csv' AS line  MERGE (s:StoreName {StoreName: line.store\_name})  MERGE (c:City {City: line.city})  MERGE (z:Zip {Zip: line.zip\_code})  MERGE (co:County {County: line.county})  MERGE (cn:CategoryName {CategoryName: line.category\_name})  MERGE (v:VendorName {VendorName: line.vendor\_name})  MERGE (id:ItemDescription {ItemDescription: line.item\_description, SaleDollars: line.sale\_dollars})  //Sale Relationships  MERGE (s)-[:PURCHASED\_FROM]->(v)  MERGE (s)-[:BOUGHT]->(cn)  MERGE (cn)-[:SUB\_ITEM]->(id)  MERGE (id)-[:SALE\_ID]->(i)  MERGE (v)-[:SUPPLIED]->(cn)  MERGE (v)-[:SUPPLIED]->(i)  //Location Relationships  MERGE (s)-[:IN\_ZIP]->(z)  MERGE (z)-[:IN\_CITY]->(c)  MERGE (c)-[:IN\_COUNTY]->(co) Data Load Now that we have our data in our database, we can begin exploratory queries and algorithms. | | | |
|  |  | |  |

|  |  |
| --- | --- |
|  |  |
| Graph Queries What Whiskey Categories Currently Sell the Most in Iowa?  match(i:ItemDescription)<-[si:SUB\_ITEM]- (c:CategoryName)  return  c.CategoryName as CategoryName,  sum(tointeger(i.SaleDollars)) as TotalSales  order by sum(tointeger(i.SaleDollars)) desc    When we run the above, we see that Irish Whiskies (the style we make) is the sixth most popular category in Iowa with total sales of $2,914,654 YTD. This tells us there is a small market in Iowa right now so we will need to advertise how Irish Whiskies are different/similar to the more popular categories and try and educate potential customers.  What Are the Best Selling Irish Whiskies?  match(i:ItemDescription)<-[si:SUB\_ITEM]- (c:CategoryName{CategoryName:"IRISH WHISKIES"})  return  i.ItemDescription as WhiskeyBrand,  sum(tointeger(i.SaleDollars)) as TotalSales  order by sum(tointeger(i.SaleDollars)) desc    No surprises here that Jameson is by far the biggest Irish Whiskey seller in Iowa. Accounting for over 71% of total Irish Whiskey sales. This tells us we should focus on converting this strong customer base by comparing our product to that one.  What County Buys the Most Variety of Whiskey?  match(c:County)<-[in:IN\_COUNTY]-(s:StoreName)-[b:BOUGHT]->(i:ItemDescription)  return  c.County as County,  count(b) as Whiskies,  sum(tointeger(i.SaleDollars)) as TotalSales  order by Whiskies desc    This tells us that Polk county is by far the largest purchaser of whisky. This is where the city of Des Moines is, but it is interesting that they have such a wide margin as compared to the other counties in Iowa. | |

|  |  |
| --- | --- |
|  |  |
| Graph Algorithms Now that we have a basic understanding of what types of whiskey are currently being sold in Iowa and where. We can move on to more advanced analysis.  Community Detection with Closeness Centrality & Label Propagation  The purpose of these algorithm is to cluster stores into groups based on the vendors they purchase from. From there I can see which vendors carry the most Irish style whiskeys and target those stores and vendors for our future sales channels. Unfortunately, I have not yet been able to get these to work since it needs to be in a monopartite graph, and I haven’t figured out how to transform my current k-partite graph into the appropriate form.  **Closeness Centrality**  CALL gds.alpha.closeness.stream({  nodeProjection: "VendorName",  relationshipProjection: "PURCHASED\_FROM"  })  YIELD nodeId, centrality  RETURN gds.util.asNode(nodeId).VendorName, centrality  order by centrality desc    **Label Propagation**  call gds.labelPropagation.stream({  nodeProjection: "StoreName",  relationshipProjection:"PURCHASED\_FROM",  maxIterations:50  })  yield nodeId, communityId  return communityId as label,  collect(gds.util.asNode(nodeId).StoreName) as StoreName  order by size(StoreName) desc    **Page Rank**  The intention of this algorithm is to understand which vendors have the biggest impact on overall sales to help narrow in on the ones we should target to distribute our whiskey. Unfortunately, I have not yet been able to get this to work since it needs to be in a monopartite graph, and I haven’t figured out how to transform my current k-partite graph into the appropriate form.  call gds.pageRank.stream({  nodeProjection: "VendorName",  relationshipProjection:"PURCHASED\_FROM",  maxIterations:20,  dampingFactor:0.85  })  YIELD nodeId, score  return gds.util.asNode(nodeId).VendorName as Vendor, score  order by score desc | |
|  | |