

---

# ***VOORAY***

---

**MIS 5900  
King of Lions  
December 5, 2019**

---

**Prepared by King of Lions Consulting:**

Grant Brinkerhoff

Cody Crofoot

Henry Olsen

Jared Robinson

---





---

TABLE OF CONTENTS

---

<b>CONTACT INFORMATION</b>	<b>6</b>
<b>GOAL STATEMENT</b>	<b>6</b>
<b>SYSTEMS DESIGN AND REQUEST</b>	<b>7</b>
PROJECT SPONSOR	7
BUSINESS NEED	7
BUSINESS REQUIREMENTS	7
BUSINESS VALUE	7
SPECIAL ISSUES AND CONSTRAINTS	7
<b>FEASIBILITY ANALYSIS</b>	<b>8</b>
TECHNICAL FEASIBILITY	8
ECONOMIC FEASIBILITY	8
ORGANIZATIONAL FEASIBILITY	9
<b>FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS</b>	<b>10</b>
FUNCTIONAL REQUIREMENTS	10
NON-FUNCTIONAL REQUIREMENTS	10
<b>CLIENT COMMUNICATION</b>	<b>11</b>
TRANSCRIPT OF MEETINGS	11
9/12/19	11
9/18/19	11
TEXT MESSAGES	12
9/4/19	12
9/5/19	13
9/10/19	13
9/11/19	13
9/12/19	13
9/18/19	13
OTHER CLIENT COMMUNICATIONS	13
<b>STAKEHOLDER REPORT</b>	<b>15</b>
<b>MEETING MINUTES</b>	<b>16</b>
9/19/19	16
9/19/19	16
9/5/19	16
9/9/19	17
9/12/19	18
9/17/19	18
<b>GANTT CHART AND ASSUMPTIONS</b>	<b>19</b>
ASSUMPTIONS	19
<b>USE CASES</b>	<b>22</b>
<b>ACTIVITY DIAGRAM</b>	<b>27</b>



<b>ER DIAGRAM/ CLASS DIAGRAM</b>	<b>28</b>
<b>DESIGN ALTERNATIVES FEASIBILITY ANALYSIS</b>	<b>29</b>
SUMMARY	29
<b>UI DESIGN</b>	<b>30</b>
INITIAL DIALOG BOX	30
WINDOW SHOWING OPERATION OF SCRIPT	30
WINDOW SHOWING SCRIPT HAS EXECUTED	31
EXAMPLE OF VISUALIZATION PROGRAM USAGE	31
<b>WINDOW NAVIGATION DIAGRAM</b>	<b>32</b>
<b>HARDWARE AND SOFTWARE SPECIFICATIONS</b>	<b>33</b>
<b>COMMUNICATION WITH CLIENT</b>	<b>33</b>
TRANSCRIPT OF MEETINGS	33
10/15/19	33
TEXT MESSAGES	34
10/14/19	34
10/16/19	34
OTHER CLIENT COMMUNICATION	34
<b>MEETING MINUTES</b>	<b>35</b>
9/19/19	35
9/23/19	35
9/26/19	35
9/30/19	36
10/3/19	36
10/7/19	36
10/10/19	37
10/14/19	37
10/17/19	38
10/21/19	38
10/24/19	39
<b>STAKEHOLDER REPORT</b>	<b>40</b>
<b>GANTT CHART AND ASSUMPTIONS</b>	<b>41</b>
ASSUMPTIONS	41
<b>IMPLEMENTATION PLAN</b>	<b>44</b>
POTENTIAL ISSUES	44
<b>TEST PLANS AND TEST RESULTS</b>	<b>45</b>
<b>DOCUMENTATION</b>	<b>49</b>
DOCUMENTATION FOR END USERS	49
<i>Running the Script:</i>	49
<i>USING THE VISUALIZATIONS</i>	50
DOCUMENTATION FOR TECHNICAL/ MAINTENANCE	51
<i>Set Up:</i>	51
<i>Setting up the database:</i>	52



<i>Connecting to Power Bi:</i>	54
<i>Code Overview:</i>	57
Create_db_c.py Overview:	57
Script_c.py Overview:	57
Drop_db_c.py Overview:	59
<b>COMMENTED CODE</b>	<b>60</b>
CREATION OF DATABASE	60
DROP DATABASE	62
DATABASE POPULATION SCRIPT (COMMENTED)	63
<b>CORRECTED DIAGRAMS</b>	<b>72</b>
UPDATED USE CASES	72
UPDATED ACTIVITY DIAGRAM	76
UPDATED WINDOWS NAVIGATION	76
<b>COMMUNICATION WITH CLIENT</b>	<b>77</b>
TRANSCRIPT OF MEETINGS	77
TEXT MESSAGES	77
10/19/19	77
11/19/19	77
11/19/19	78
11/29/19	78
OTHER CLIENT COMMUNICATION	78
<b>MEETING MINUTES</b>	<b>79</b>
10/24/19	79
10/28/19	79
10/31/19	80
11/04/19	80
11/07/19	80
11/11/19	81
11/14/19	81
11/18/19	82
11/20/19	82
11/21/19	82
11/25/19	83
12/2/19	83
12/4/19	84
<b>STAKEHOLDER REPORTS</b>	<b>85</b>
<b>GANTT CHART</b>	<b>86</b>



# DELIVERABLE 1

---

9/19/18



## CONTACT INFORMATION

---

Grant Brinkerhoff	(801) 888 - 5811	Grantbrinkerhoff@gmail.com
Cody Crofoot	(208) 313 - 2572	Cdcrofoot374@gmail.com
Henry Olsen	(435) 704 - 2315	Henry.olsen@aggiemail.usu.edu
Jared Robinson	(669) 251 - 9597	Jared.colin.robinson@gmail.com

## GOAL STATEMENT

---

*"TO CREATE A SYSTEM TO PERFORM DATA  
ORGANIZATION AND VISUALIZATION FOR VOORAY  
WITH LITTLE TO NO COST FROM THE CLIENT  
COMPLETED BY DECEMBER 5TH, 2019."*

---



## SYSTEMS DESIGN AND REQUEST

---

### PROJECT SPONSOR

---

**Brad Rigby**

*Director of Finance at Vooray*

**Email:** [brad.rigby@vooray.com](mailto:brad.rigby@vooray.com)

**Phone Number:** (435) 881-6133



### BUSINESS NEED

---

The current process for gathering Vooray sales from Amazon and setting them up for visualization for analysis is a time-consuming process that must be done manually every day when sales are updated.

### BUSINESS REQUIREMENTS

---

- The system will be able to upload data from an excel spreadsheet into a database while maintaining data integrity.
- The system will run stored procedures that we will create in the database to organize and pivot data in preparation for data visualization.
- The system will automatically update visualization information through connection to the database to allow Vooray to have a dashboard of information to perform analysis on.

### BUSINESS VALUE

---

Time will not have to be spent gathering, organizing, and visualizing data from Amazon so more time can be spent on the analysis and planning for improvement. All the clients will have to do is download their spreadsheet of daily sales they want to perform analysis on and run our software.

### SPECIAL ISSUES AND CONSTRAINTS

---

Not all team members are familiar with python and will have to learn how to use it to develop the system as well as take time to either learn or refresh their SQL abilities.



## FEASIBILITY ANALYSIS

---

### TECHNICAL FEASIBILITY

---

#### *Familiarity with Functional Area (low Risk)*

- Grant Brinkerhoff has a positive relationship with the client. He has a strong background in finance which will be useful in helping to build strong reports that the client will be able to utilize.

#### *Familiarity with Technology (medium risk)*

- Cody Crofoot and Henry Olsen have background writing python programs to pull data and put it into a database. Henry has a solid SQL background and can help with setting up the database. Jared and Grant have experience building dashboards and visualizations which can be utilized. No one in the group has any experience with pulling data from Amazon and there is risk depending on how complicated it becomes.
  - Cody will focus on the python programming aspects of the program and Henry will work with setting up the database to house the information. Jared and Grant will use their skills to use that data to create customizable dashboards for the client.

#### *Project Size (low risk)*

- Our team has been in good communication on the extent of our abilities and time. This has resulted in a project that we feel is very feasible for our constraints.

#### *Compatibility (medium risk)*

- The biggest unknown in the project is how to use python to pull the Amazon data for the client. We are looking to create a program that runs automatically but if that is not feasible the client is ok with a manual program as well.

### ECONOMIC FEASIBILITY

---

#### Tangible Costs

- No cost for labor for the production of the system by the client
- For the database, we will be using MySQL server which is an open-source relational database that does not have a licensing cost
- The rest of the system will be developed by python which is a free download online

#### Tangible Benefits





- The client would have savings of more than \$17.5K during the usage of the system.<sup>1</sup>
- Time-saving of more than 5 hours a week.

#### Intangible Benefits

- Reduced time is taken gathering, organizing, and visualizing data so more time can be spent analyzing the data instead
- Allows easy access to data visualization for any employee rather than just those who know how to organize the data

#### Intangible Costs

- Time is taken by the group to create a system
- Time is taken for some group members to become familiar with python and refresh themselves on SQL

---

### ORGANIZATIONAL FEASIBILITY

---

- This system will be appreciated and used by the team at Vooray as instead of manually having to organize their data and set up their visualization it will only take them a few clicks to run the system and get it done.
- The system will be designed to be very easy to use and not require maintenance unless amazon makes major changes to their Excel data output in which only a basic understanding of SQL/PowerBI would be required to update the system.

---

<sup>1</sup> Savings were calculated by taking the Net Present Value (NPV) of the hourly wages saved over 5 years. Wages were estimated at last known cost of \$20/hour and increase 5% yearly. There were 48 work weeks a year, and 40 hours for each week. NPV was discounted by an estimated WACC of 15%.



## FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

---

### FUNCTIONAL REQUIREMENTS

---

- **Data Collection**
  - Performs batch operations on a set schedule
  - Retrieve information from Amazon Seller Workspace
  - Formats and cleans data according to specifications
- **Database**
  - Creates tables for inventory and all orders reports
  - Updates automatically with most current information
- **Dashboards**
  - Allows the client to drill down by specific product
  - Visualizes trends in inventory metrics
  - Illustrates sales data in meaningful charts
  - Provide real-time analysis on key metrics

### NON-FUNCTIONAL REQUIREMENTS

---

- **Operational**
  - The system will enable our client to focus their time on analysis and improvement.
  - The system will be accessed through a PowerBI dashboard.
- **Performance**
  - The system will update the data daily.
- **Security**
  - Only our client will have access to our system.
- **Political/Cultural**
  - No requirements necessary



## CLIENT COMMUNICATION

---

### TRANSCRIPT OF MEETINGS

---

---

9/12/19

---

**Time:** 4:45 pm

**Location:** Huntsman Hall 376

**In Attendance:** Grant Brinkerhoff, Cody Crofoot, Brad Rigby

#### Problem #1

One of the largest issues is that with B2B there is a lot of prepayment. The company will bill before the shipment, but the problem is that they ship after they bill. What they really want is to better estimate what the shipping will be. Brad has an excel form that basically has all of their products and all of their sizes. This form can be updated with the information from UPS but it has to be done manually. What Brad would like us to do, is to be able to update that information automatically. That way it does not need to be manually updated and the sheet can be an automatic predictor of shipping costs.

#### Problem #2

Vooray also sales a lot via Amazon. This information has each product, how much it was sold for each day, how much product was in inventory, and how much product was moved/sold. Brad is wondering if we can have something that is able to pull that information, transform it, and then visualize it or make it useful to the company.

---

9/18/19

---

**Time:** 5:00 pm

**Location:** Remote Phone Call

**In Attendance:** Cody Crofoot

This was a phone call to go over the current process with Brad to understand the pain points. The following is his explanation of the current process.



The Process As-Is:

Goes onto Amazon Seller Central

Manually downloads three different reports (.CSV file)

Daily Inventory History Report

Daily inv level

Quantity Sold that Day

Price Sold For

All Orders

(filters by date and adds on to ongoing)

Gives a history of every order had

Filters to quant sold, price

Sticks into excel Sheet

Has to create a pivot table to get data in

Then creates VLOOKUP to get into it into a single sheet

Then has to create the visualization

then has to scroll through the graphs

Wants to Use PowerBI

Filter the graph from there

---

## TEXT MESSAGES

---

---

9/4/19

---

**Grant:** Hey Brad. Do you remember how I told you about the MIS capstone class last year? Did Vooray have anything that they needed potentially automated? Like a database, excel project, visualization, etc.? If not that's okay.

**Brad:** Hey! I actually think we might have a couple of different ideas here. One question is if you guys would know how to connect to an API? For example, we have a software we use, and there is an open API where we could connect Excel or PowerBI to pull in reports automatically. If not, that's okay, I might have another project that I can explain to you some time.

**Grant:** Yeah, I know that I don't know how much about APIs, but I think that the others in my group do.

**Brad:** Cool! Yeah, maybe we could meet sometime this week and I can explain some of the different ideas.

**Grant:** Yeah, definitely. How available are you tomorrow?

**Brad:** Yeah, I will have some time tomorrow. I'll be up on campus on Tuesdays and Thursdays from 1:30 to 3 and 4:30 to 5.

**Grant:** Let's try for tomorrow at 4:30, and if not then let's try for Tuesday.



---

9/5/19

**Grant:** Hey Brad, I can't make it today can we try for some time on Tuesday?

**Brad:** It's okay! Sorry, I'm busy as well. Let's try that. If you don't have a class and want to come into my work tomorrow, then you can do that.

**Grant:** I'll be in Salt Lake for UGF, but I'll forward you some questions.

---

9/10/19

**Brad:** Hey Grant can we meet up on Thursday?

**Grant:** Yeah, we can do that. A couple of questions that the group wanted to ask you for Thursday. I thought it would be best for you to know them and don't have to answer them off the top of your head.

What Framework is Vooray using?

What is going to happen with the data from the API?

Does the Data need to be altered or visualized? If so, how?

---

9/11/19

**Brad:** Perfect! Yep, these are all questions that I can answer. It will be good to talk through them tomorrow with you.

---

9/12/19

**Brad:** I can meet any time before 3.

**Grant:** Awesome. I'll be there in about 5 minutes. I've got a study room booked on the 3rd floor.

**Brad:** Sounds good.

---

9/18/19

**Cody:** Hey Brad this is Cody Crofoot. We met last week, I am in the MIS capstone class and my group is helping to build an application for your company. I was wondering if you were available to meet sometime today briefly in person or over the phone. I just had some quick had some quick questions for you as we are preparing our plan.

**Brad:** Hey Cody, sorry this is late. Today has been a busy day thanks for reaching out. I can take a call this afternoon or we can meet in person at the same time as last time tomorrow. Let me know what works. Thanks!

**Cody:** Hey Brad I am in class until 4:45 but I can call after if that works

**Brad:** Yeah, that works.

---

## OTHER CLIENT COMMUNICATIONS

---



## Amazon Report

1 message

Bradly Rigby <brad.rigby@vooray.com>  
To: cdcrobot374@gmail.com

Wed Sep 18, 2019 at 5:20 PM

### Daily Inventory Report

asin	sku	product-name	quantity	fulfillment-center-id	detailed-allocation	country
------	-----	--------------	----------	-----------------------	---------------------	---------

I just summed the Quantity by sku on the same day because there are multiple fulfillment centers with the same sku.

amazon-id	amazon-asin	product-name	quantity	fulfillment-center-id	detailed-allocation	country
-----------	-------------	--------------	----------	-----------------------	---------------------	---------

Let me know if you have any questions.

Thanks!

**VOORAY**

*Brad Rigby*

Office: 435.535.3665

brad.rigby@vooray.com



## STAKEHOLDER REPORT

---

Date: September 12, 2019  
To: Brad Rigby, Vooray Director of Finance  
From: King of Lions Design Team  
Subject: Creation of Sale Database and Visualization

Dear Mr. Rigby,

We are very appreciative and excited about the chance to help create a new database system and visualization tools for Vooray's Amazon Sales. We are hoping that we will be able to meet your needs and create a product that will help you in the years to come.

We have started to put together a large part of your project and have completed much work on the outset steps. We have developed a report that contains the business requirements, requirements of what the software needs to do (or its functional requirements), how the software will do it (or the non-functional requirements), and analysis of the feasibility of Vooray's operations, economic, and technical situation.

Your program will allow a download of Vooray's sales from Amazon's selling services, an upload and subsequent transformation of the data into a star schema for a MYSQL, or MYSQL similar, database and finally, a PowerBI integration of the information that will be able to visualize the data and turn it into actionable information for you and Vooray to use. We want to be able to make this process as streamlined and simple as possible for you to be able to run.

Brad, we also want to keep you involved and as up to date as possible. We hope that as we keep in constant contact with you so that you may have a good understanding of our process and where we are.

Please let us know if there is anything that we can do for you.

Sincerely,  
The King of Lions Consulting Team



## MEETING MINUTES

---

9/19/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, and Jared. Henry out sick (Informed us)

**Minutes:** The group was made. The team contract was talked through and then created. Meeting times were set forth and there was a discussion on the future project.

### Action Items

Everyone: Get Clients  
Cody: Talk to his wife about graphic design

9/19/19

---

**Time:** 11:00 am  
**Location:** Huntsman Hall 274  
**In attendance:** Grant, Cody, Jared, and Henry (on the phone)

**Minutes:** We first finished the discussion with Henry and what his skills are. He is able to do really well in SQL and data analytics.

We then started to go through the potential projects that we could be working on.

### Action Items

Grant: US Courts, Vooray, and Paul Fjeldsted  
Cody: Stokes Nature Center and a couple of small businesses  
Jared: Cache Valley VA Association and Logan City  
Henry: Will reach out, especially when he is feeling better.

9/5/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Jared, and Henry. Cody out (Informed us)





**Minutes:** The reviewed action plan from the last meeting, Grants Vooray Client (Brad) reached out that we could help him out with some projects that they need to be done in relation to API, Jared had some follow-up from Cache Valley Veterans Administration (CVVA) but no specifics on a project.

Grant reached back out to Brad that we would be interested in helping assist him and requested for specifics of what they needed assistance with, generated team name and turned in client info deliverable

### **Action Items**

Cody: Will start looking at coding requirements for APIs and talk to his wife about designing a simple logo for us.

Grant: Reach out to Brad and try to set up a time where we can meet and come up with a good description of the problem and their current technologies.

Henry: Will start looking at the general requirements for APIs

- What web framework is Vooray using?
- Looked at an example of how you can grab data from an API.
- Looked at a tutorial of how to build a REST API with Python using Flask.

Jared: Will start looking at designing the system requests section.

9/9/19

---

**Time:** 11:00 am

**Location:** Huntsman Hall 274

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** Grant has set up a meeting with Brad for Tuesday, September 10th, 2019. Cody will try to be in attendance and we will just ask him questions that we come up with today.

Questions to ask Brad:

- What web framework is Vooray using?
- What do you do with the data from the API?

### **Action Items**

Grant: Email out the template for the memo. Add beginning details for it. Meet with Brad. Keep notes over meetings.

Cody: Look into the technical specifics of what we need to ask Brad. Ask wife about Logo.

Henry: Start looking into Microsoft Project and looking at putting together the Gantt Chart.

Jared: Finish system-request before Thursday.



Thursday Meeting: Look at feasibility analysis. Create an action plan for finishing D1.

9/12/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** Followed up on meeting notes from Grant and Cody's meeting with Brad about the finalized project description. Decided on amazon tracking streamline the process, the customer needed their data from amazon sales exported from amazon which comes in an excel format, transferred into a database, pivoted and organized, then transferred to a visualization software (ex. Microsoft PowerBI) as a dashboard for analysis for Vooray to see.

Worked on finalizing "goal statement" and "system request" and discussed action items to prepare for a Tuesday night meeting where we will prepare for our presentation.

**Action Items**

Cody: Finish technical feasibility analysis, Function requirements  
Grant: Stakeholder report, client communication  
Henry: Gantt Chart, Non-functional requirements  
Jared Robinson: Organizational and Financial feasibility analysis.

9/17/19

---

**Time:** 4:30 pm  
**Location:** Eccles Business Building 120  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** The group met together so that we could try to make sure that we had all aspects of our project under control. We made sure that we knew the requirements for the first deliverable and then started working on piecing together parts of our sections. Action items were going to stay the same for this week, and then we are going to plan for a meeting at 2:00 pm and at 4:30 pm on Thursday, September 19<sup>th</sup>.

**Action Items**

Cody: Finish technical feasibility analysis, Function requirements, start editing the deck  
Grant: Stakeholder report, client communication, start editing the memo  
Henry: Gantt Chart, Non-functional requirements  
Jared Robinson: Organizational and Financial feasibility analysis



## GANTT CHART<sup>2</sup> AND ASSUMPTIONS

---

### ASSUMPTIONS

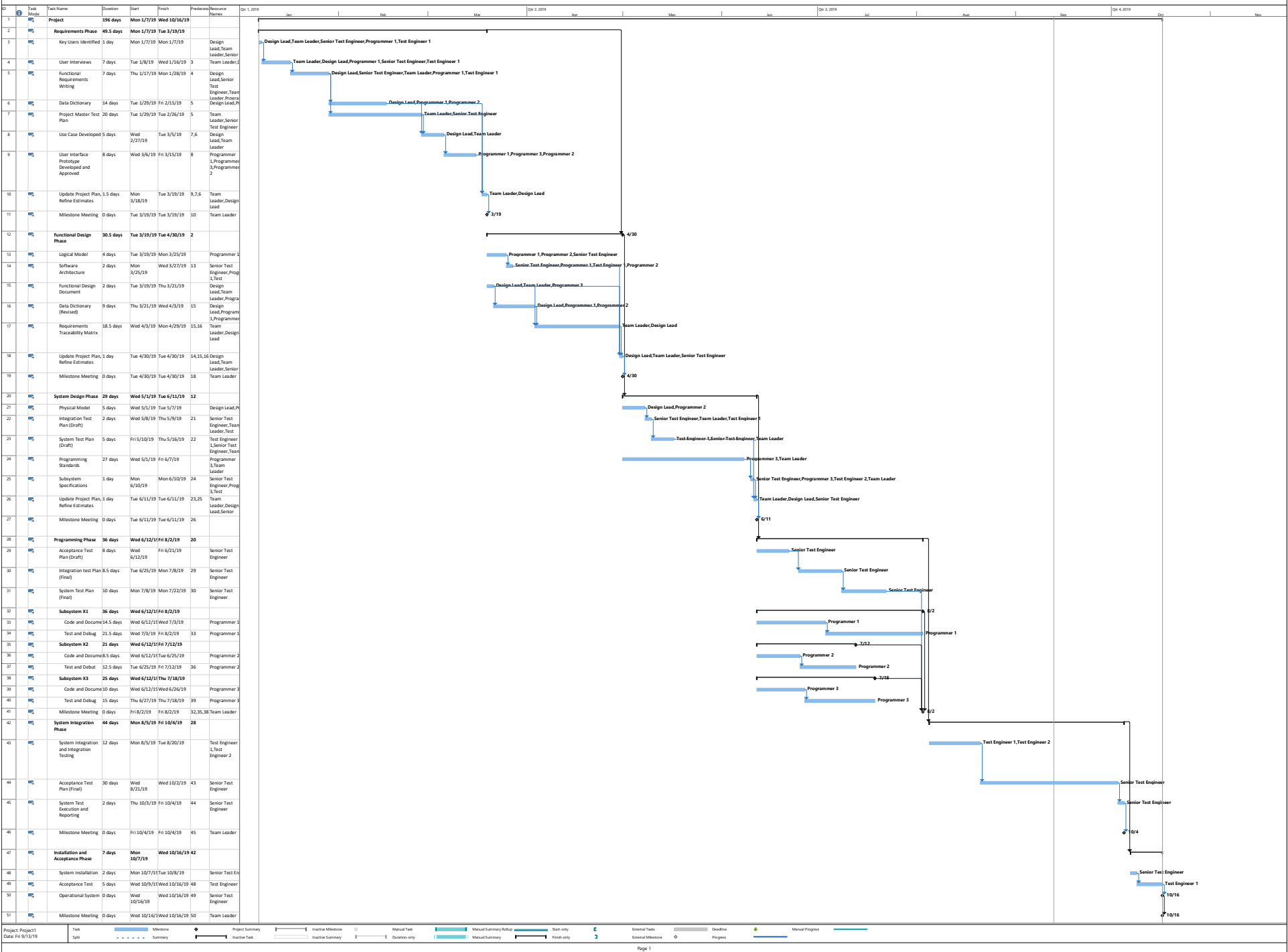
---

- Splitting up our tasks will enable us to meet our deliverable deadlines.
- Similar tasks in deliverable 1 will be assigned to the same team members in deliverables 2 and 3.

---

<sup>2</sup> Gantt chart is subject to change depending on the length of task, skill set, and the needs of our client.

Figure 1: Task List and Gantt Chart





## DELIVERABLE 2

---

10/24/17



## USE CASES

<b>Use Case Name:</b> Download Data	<b>ID:</b> 001	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Essential	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The user goes to amazon and downloads the sales they would like cleansed and uploaded to the database		
<b>Trigger:</b> User wants to initiate the process  <b>Type:</b> External		
Relationships: Association: Include: Extend: Generalization:		
<b>Normal Flow of Events:</b>  <ol style="list-style-type: none"><li>1. Login to Amazon Account</li><li>2. Go to order history reports in Your Account</li><li>3. Select the report type from the drop-down menu, then fill in the start date, end date, and report name. Click Request Report.</li><li>4. When the report is complete, Amazon will send/receive an e-mail notification. To retrieve the report, visit Order History Reports and click Download.</li><li>5. Move data CSV file to Project_Folder</li></ol>		
SubFlows:		
Alternate/Exceptional Flows:		



<b>Use Case Name:</b> Clean Data	<b>ID:</b> 002	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, essential	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The user downloads information and report off of Amazon's Seller Central Sales System and runs the script.		
<b>Trigger:</b> User runs the application from the folder Type: External		
<b>Relationships:</b> Association: Include: Extend: Load Data Generalization:		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"><li>1. The user initiates our application</li><li>2. The user selects Cleans Data option</li><li>3. Dates are changed to date format from dateTime format</li><li>4. Removes any CUSTOMER_DAMAGED rows from the data</li><li>5. Updates any Amazon.ca cells to Amazon.com</li><li>6. Removes any rows with dates older than 30 days ago</li><li>7. Removes any rows without a price or quantity</li><li>8. Saves cleansed file</li></ol>		
Sub Flows:		
Alternate/Exceptional Flows:		



<b>Use Case Name:</b> Load Data	<b>ID:</b> 003	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Essential	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> Data in CSV file that was already cleansed is uploaded to the database		
<b>Trigger:</b> Data extraction is complete <b>Type:</b> Internal		
<b>Relationships:</b> Association: Include: Load Data Extend: Generalization:		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"><li>1. If the user is going through process manually then they would select the Load Data button from the application</li><li>2. The script opens MySQL application on the computer</li><li>3. The script runs SQL stored procedure to pull data from the CSV file and insert it into the database</li></ol>		
SubFlows:		
Alternate/Exceptional Flows:		





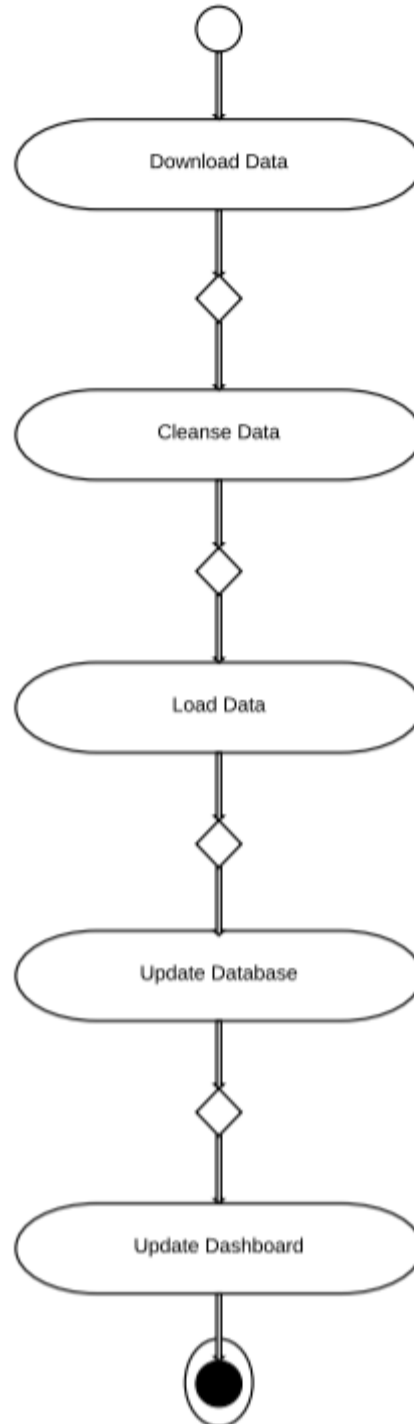
<b>Use Case Name:</b> Organize Database	<b>ID:</b> 004	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> Data is loaded into the database and pivoted per request of the client for analysis he desires		
<b>Trigger:</b> Script finishes altering CSV file Type: Internal		
<b>Relationships:</b> Association: Include: Extend: Organize Dashboard Generalization:		
<b>Normal Flow of Events:</b>  1. If the user is going through process manually then they would select the Update Database button from the application  2. The script initiates stored procedure to pivot quantity by SKU (which is an identifier column)		
SubFlows:		
Alternate/Exceptional Flows:		



<b>Use Case Name:</b> Update Dashboard	<b>ID:</b> 005	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The dashboard is updated with the new data		
<b>Trigger:</b> Script finishes altering CSV file Type: Internal		
<b>Relationships:</b> Association: Include: Extend: Generalization:		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"><li>1. If the user is going through process manually then they would select the Update Dashboard button from the application</li><li>2. The script then refreshes the PowerBI dashboard linked to the database which has already been set up to show the number of sales by date also sums the quantity of all sales</li></ol>		
SubFlows:		
Alternate/Exceptional Flows:		



## ACTIVITY DIAGRAM

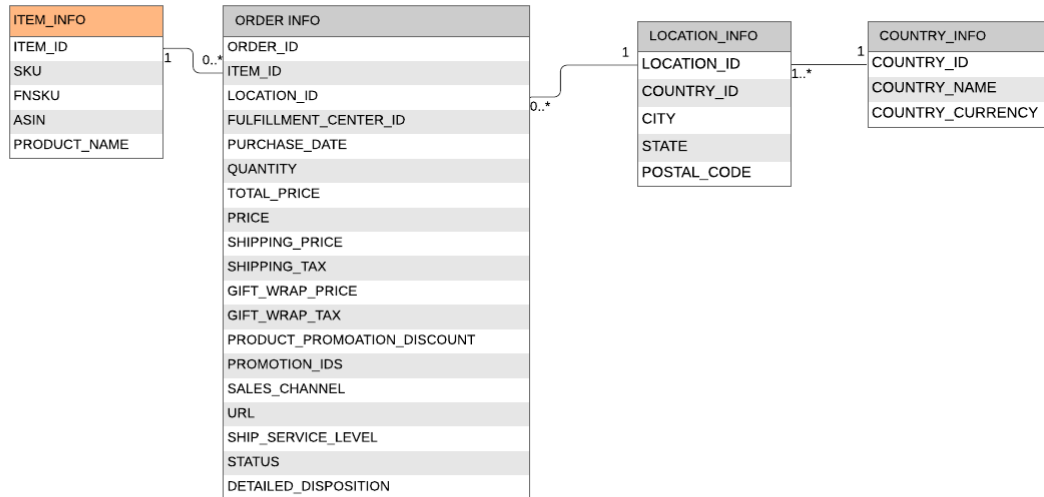




## ER DIAGRAM/ CLASS DIAGRAM

### King of Lions ERD

Henry Olsen | October 23, 2019





## DESIGN ALTERNATIVES FEASIBILITY ANALYSIS

Alternative Matrix	Alternative 1 - Python/SQL	Alternative 2 - Excel/SQL	Alternative 3 - Outsource through Upwork
Technical Feasibility	All in the group have taken both database classes and are familiar with SQL. However, only about half the group has experience using python.	All in the group have taken both database classes and are familiar with SQL. No one in the group is proficient with VBA	The project would be outsourced to professionals with the necessary skillset
Economic Feasibility	Python is a free open source software with no cost. The SQL software necessary is free.	The client already uses excel; The SQL software necessary is free.	Prices vary according to each individual but are around \$75/hr.
Organizational Feasibility	The customer would not be required to know Python or SQL to run the program and no maintenance on the system is projected.	Customer is already familiar with Excel and would not be required to know SQL	The organization is not looking to hire someone to fix this problem
Other Benefits	One easy script to run the client through the entire process of updating the dashboard by just clicking one button.	Customer is familiar with excel and would be able to integrate with the current system	Professional would be able to implement unique ideas gained from experience
Other Limitations	Lack of experience in using the Amazon API to automatically download the data.	Excel potentially will not be able to handle data as well as other methods	Customer is not looking to hire out for this project

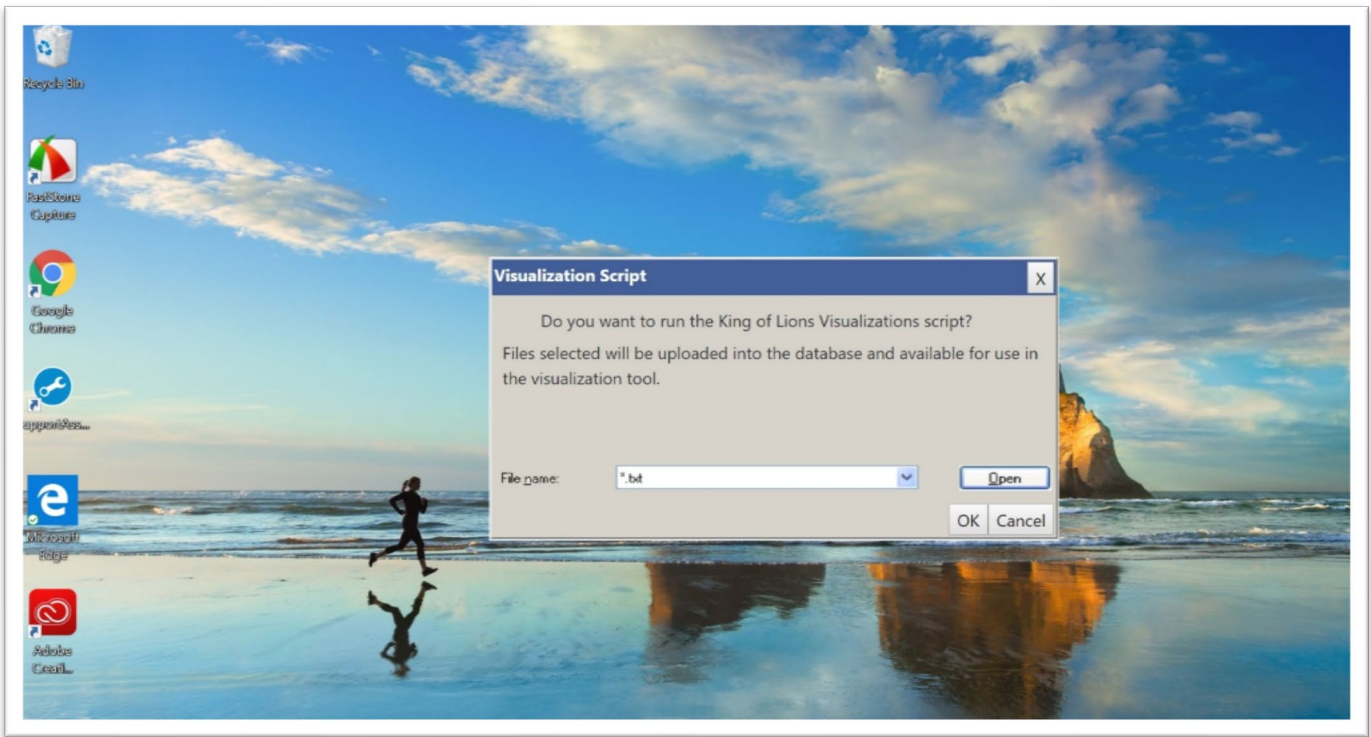
## SUMMARY

*Our team will be using the Python/SQL approach. We feel that the two different software are the best suited to help the company and are well suited to the skill set of the group.*

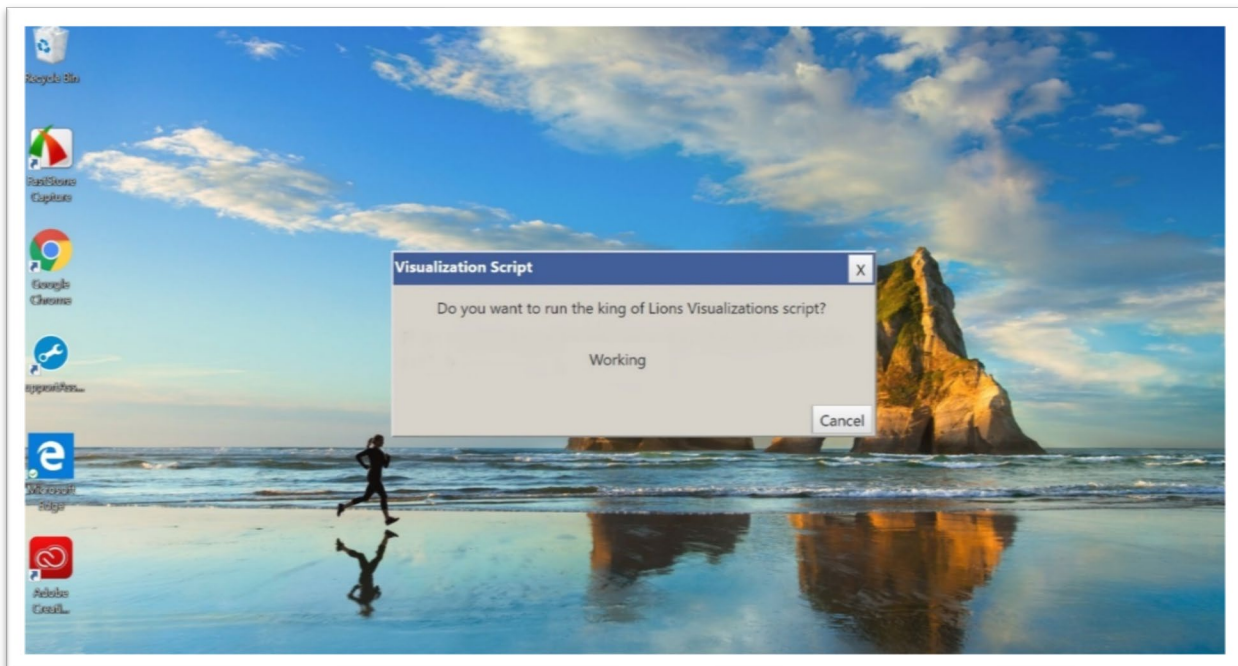


## UI DESIGN

### INITIAL DIALOG BOX

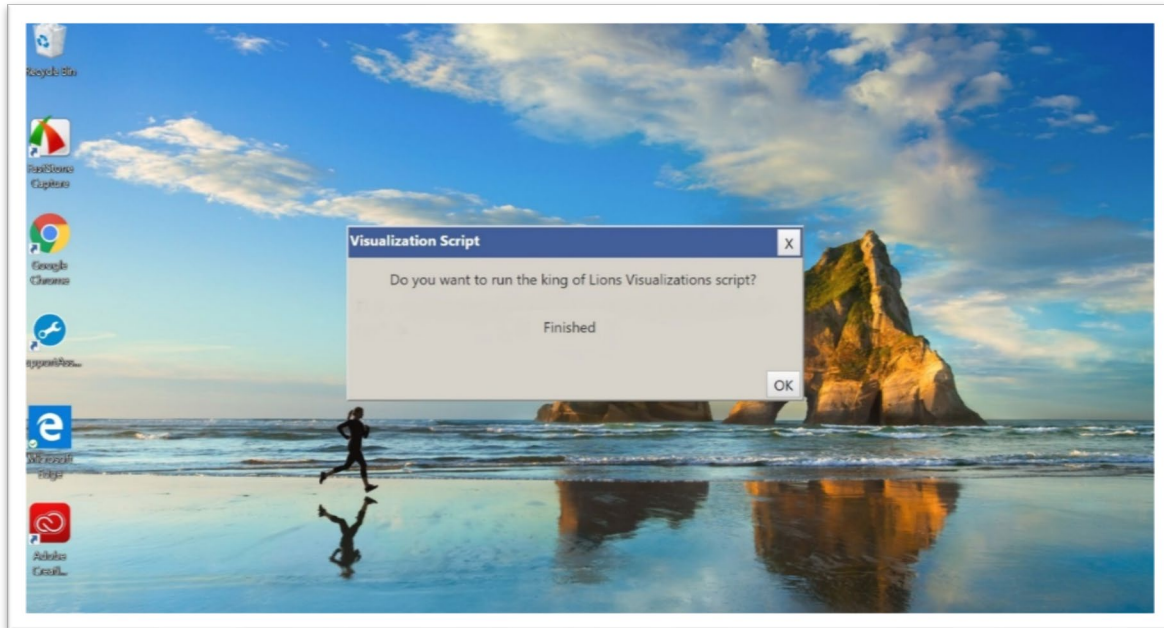


### WINDOW SHOWING OPERATION OF SCRIPT

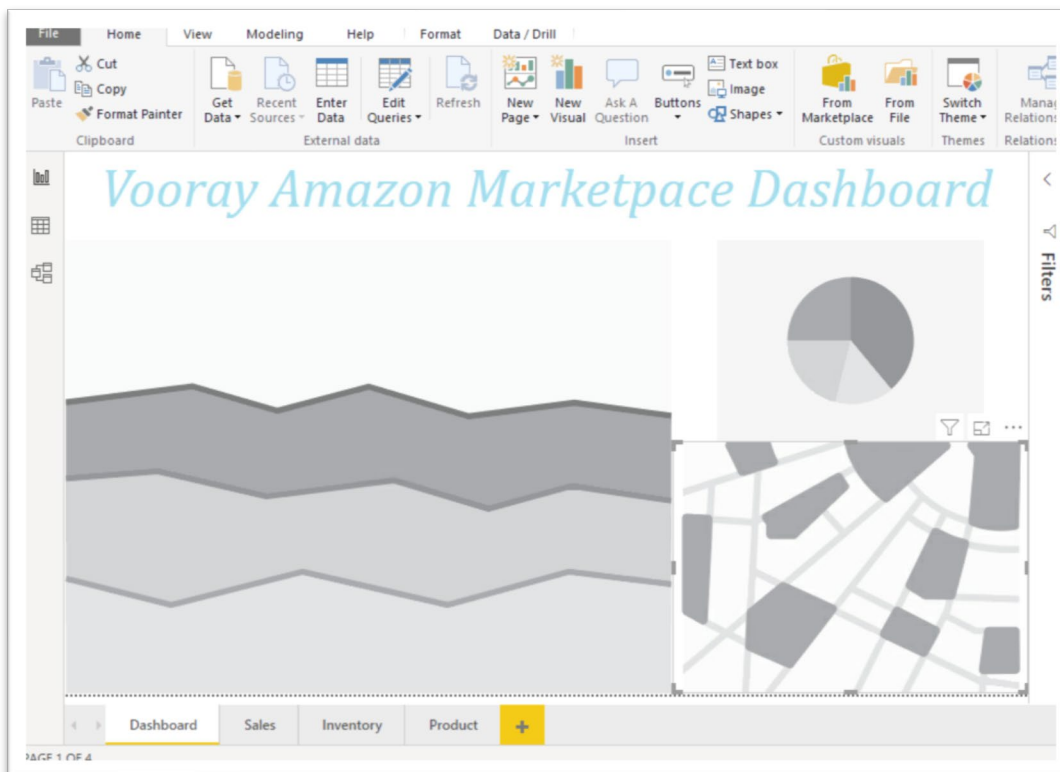




## WINDOW SHOWING SCRIPT HAS EXECUTED



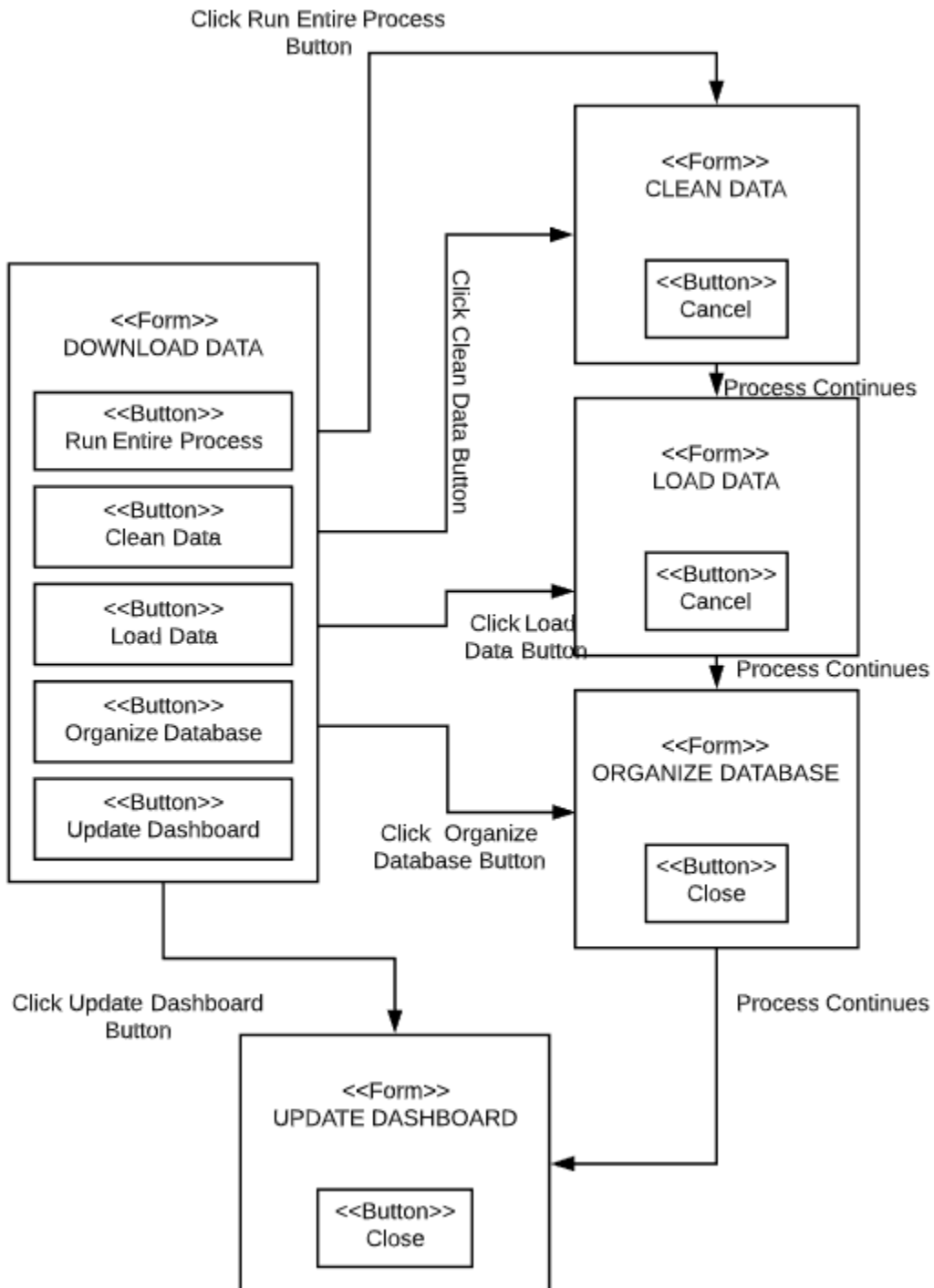
## EXAMPLE OF VISUALIZATION PROGRAM USAGE







## WINDOW NAVIGATION DIAGRAM







## HARDWARE AND SOFTWARE SPECIFICATIONS

Operating System	Windows Mac
Software	MySQL PowerBI Python
Hardware	PowerBI recommended 1 GB Ram MySQL recommended 4 GB Ram  Data requirements are dependent upon how much data the client wants to store in the database at any given time
Network	Broadband Ethernet

## COMMUNICATION WITH CLIENT

### TRANSCRIPT OF MEETINGS

10/15/19

**Time:** 4:00 pm

**Location:** Remote Phone Call

**In Attendance:** Cody Crofoot, Grant Brinkerhoff

1. Vooray is going to be to an Oracle Netscape Database by December. The go-live date is November 15, and Oracle is helping them build the database to their needs. We might be able to build a partition for us.

2. Reports that need to be pulled

**Daily Inventory**

- Set the dates
- Time doesn't matter, only the dates matter
- Sum the Quantities
  - Doesn't need to be broken out by the warehouses, just the Product SKU
- Automatically takes out defective and warehouse damaged items.
- Get a list of all SKUs
- Day-to-day inventory tracking
- All of these reports track day by day



-This is to see how much the prices affect the sales

### Price and Sales (All Orders Report

- Has to be every thirty days
- Needs to be able to filter out Amazon.ca
  - They only want Amazon USA
- They want to see the quantity sold by date
  - There should be a sum of the product SKUs.
- Item price
  - Again, they want to see if this will affect their sales

3. Amazons APIs are extremely hard to be able to get access to. It probably won't be possible to write a script to get access to the companies selling page, because Amazon will take too long to get the needed okay for Vooray to connect to their API.

---

### TEXT MESSAGES

---

---

10/14/19

---

**Grant:** Hey Brad, did you have time to chat about Vooray for a minute or two tomorrow? Cody and I just had some questions about what you wanted for the information pull and I was wondering what you wanted to include in your dashboard

**Brad:** Hey Grant, yeah for sure. I have time after my class. It gets out at 1:15.

**Grant:** Alright, let's do 1:20 on the 3<sup>rd</sup> floor.

**Brad:** Great, See you then.

---

10/16/19

---

**Grant:** Hey Brad, I just wanted you to know that after talking with some client support from Amazon and the forms, we might not be able to update via API

**Brad:** Hey Grant, that totally fine!

---

### OTHER CLIENT COMMUNICATION<sup>3</sup>

---

---

<sup>3</sup> No other client communication was had or needed, outside the 10/15/19 meeting and its set up.



## MEETING MINUTES

---

9/19/19

---

**Time:** 2:30 pm  
**Location:** Huntsman Hall 3<sup>rd</sup> Floor Pavilion  
**In attendance:** Cody, Grant, Henry, Jared

**Minutes:** The final deliverable was put together. As a group, we went over the memo to make sure that we were happy with where the final product and made any necessary edits. We also went over who was going to do what in the presentation.

### Action Items

Cody: Present what he worked on

Grant: Present what he worked on

Henry: Present what he worked on

Jared: Present what he worked on

9/23/19

---

**Time:** 5:15 pm  
**Location:** Remote via Google Hangouts  
**In attendance:** Cody, Grant, Henry, Jared

**Minutes:** Discussed what went well with the presentation. Begin creating a list of objectives to complete for the D2 presentation.

### Action Items

Cody: Begin drafting use cases

Grant: Start draft of memo two

Henry: Begin modifying Gantt chart

Jared: Begin drafting use cases

9/26/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Cody, Grant, Henry, Jared



**Minutes:** Worked on use case drafts as a group, discussed the execution plan for the scripts to decide how to develop use case plans and models.

**Action Items**

Cody: Begin research into visualization options

Grant: Continue formatting memo two

Henry: Begin research into the use of MySQL for the database

Jared: Begin research into other database options

9/30/19

---

**Time:** 9:00 pm

**Location:** Remote via GroupMe and Google Hangouts

**In attendance:** Cody, Grant, Henry, Jared

**Minutes:** Important information was shared about class details and what we were going to do to start preparing for the memo. Tasks were assigned out to go and look forward to the project so that we could make accurate plans.

**Action Items**

10/3/19

---

**Time:** 5:15 pm

**Location:** Huntsman Hall 124

**In attendance:** Cody, Henry, Jared. Grant out (Informed us)

**Minutes:** Went over the design of use cases and looked over the requirements for deliverable 2.

**Action Items**

Cody: Reach out to Brad about specific analytics to be done

Grant: Look into PowerBI feasibility

Henry: ER Diagram

Jared: Hardware and software requirements

10/7/19

---

**Time:** 11:30 am

**Location:** Remote via Google Hangouts



**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** Discussed progress and completions from the last meeting, reviewed list of things still needing completion. Discussed presentation ideas for D2 and how we wanted to set things up and improvements from D1.

**Action Items**

Cody: study up on APIs

Grant: setup Presentation draft

Henry: continue work on the ER diagram

Jared: Finish up hardware and software specifications

---

10/10/19

---

**Time:** 5:15 pm

**Location:** Huntsman Hall 124

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

We started working on formatting the memo for the 2<sup>nd</sup> deliverable.

**Action Items**

Cody: Have a question list for Brad ready for Monday?

Grant: Set up a meeting with Brad for Tuesday. Make sure that all minutes are updated for the memo. Finalize all use case diagrams and set them in the memo.

Henry: Start look at the information that is needed for the ER Diagram. Start looking at SQL.

Jared: Create a Lucid chart account and start a draft for the activity diagram.

---

10/14/19

---

**Time:** 7:00 pm

**Location:** Remote Call via Google Hangouts

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** Talked over the requirements for the new memo. Checked-in on last meeting action items. Set up a meeting with Brad Rigby for the following day at 1:20 pm. The purpose of that meeting is to start looking at the requirement for the dashboard that needs to be built for the project. Action items for a couple of days were set.

**Action Items**



Cody: Look into potential python modules for the script

Grant: Look into API Requirement and determine the feasibility

Henry: Continue to look into free software to utilize SQL

Jared: Finish up Lucid chard and draft of activity diagram

10/17/19

---

**Time:** 5:15 pm

**Location:** Huntsman Hall 124

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:** We went through the UI design process and started to think about how we were going to use the use cases to help with our UI Design. We started to make plans on how to make sure the memo is going to be ready by Wednesday afternoon/ Wednesday Night.

**Action Items**

Cody: Contact Brad about specific report issues

Grant: Put together memo template

Henry: ER Diagram, look over reports from Brad

Jared: Use Case review, setup GitHub account

10/21/19

---

**Time:** 5:15 pm

**Location:** Google Hangouts

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

We went through the list of everything that we have accomplished for the second deliverable. We decided that we wanted to have a majority of the deliverable done by Wednesday night so that we can focus on formatting the memo and slide deck rather than getting sections done.

**Action Items**

Cody: Create the design alternatives, create stakeholder report, help update the communications with the client, and start

Grant: Create the UI Design, create the window navigation diagram, and update meeting minutes.

Henry: ER Diagram, help answer questions about the database, and set up the Gantt Chart.

Jared: Touch up use cases, send the activity diagram to Professor Chidoba, and start looking at the Hardware and software specifications.



10/24/19

---

**Time:** 2:30 pm  
**Location:** Huntsman Hall 384  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

We spent this time going through the rest of the memo and making sure that we have everything that we wanted.

**Action Items**

Cody: Going to talk about the windows navigation and alternative matrix.  
Grant: Going to talk about client relations and UI design.  
Henry: Going to talk about the ERD Diagram, Gantt Chart, and start off the presentation.  
Jared: Going to talk about the use-cases, activity diagram, and hardware requirements.



## STAKEHOLDER REPORT

---

Date: October 21, 2019

To: Brad Rigby, Vooray Director of Finance

From: King of Lions Design Team

Subject: Project Update

Dear Mr. Rigby,

We wanted to reach out to you and give you an update on the progress of the project. We have been hard at work and are finishing up the planning stages of the project. We have gone through the technical requirements and will begin development over the next few weeks.

We have put together numerous reports detailing how the program will function. We will be utilizing Python and SQL to transform the company data and Power BI for the visualization. We have investigated other alternatives and are confident that this is the best choice.

The program will be easy to function on your end and will require little to no maintenance. Our goal is to present a deliverable that you will be able to run with minimal effort. It will be customized to develop all the reports that you have specified in previous meetings.

Brad, we will be providing you with some of the documentation for the user interface and directions for the program. If you have any questions about what you see, we will be happy to answer any questions.

Please let us know if there is anything that we can do for you.

Sincerely,

The King of Lions Consulting Team





## GANTT CHART<sup>4</sup> AND ASSUMPTIONS

---

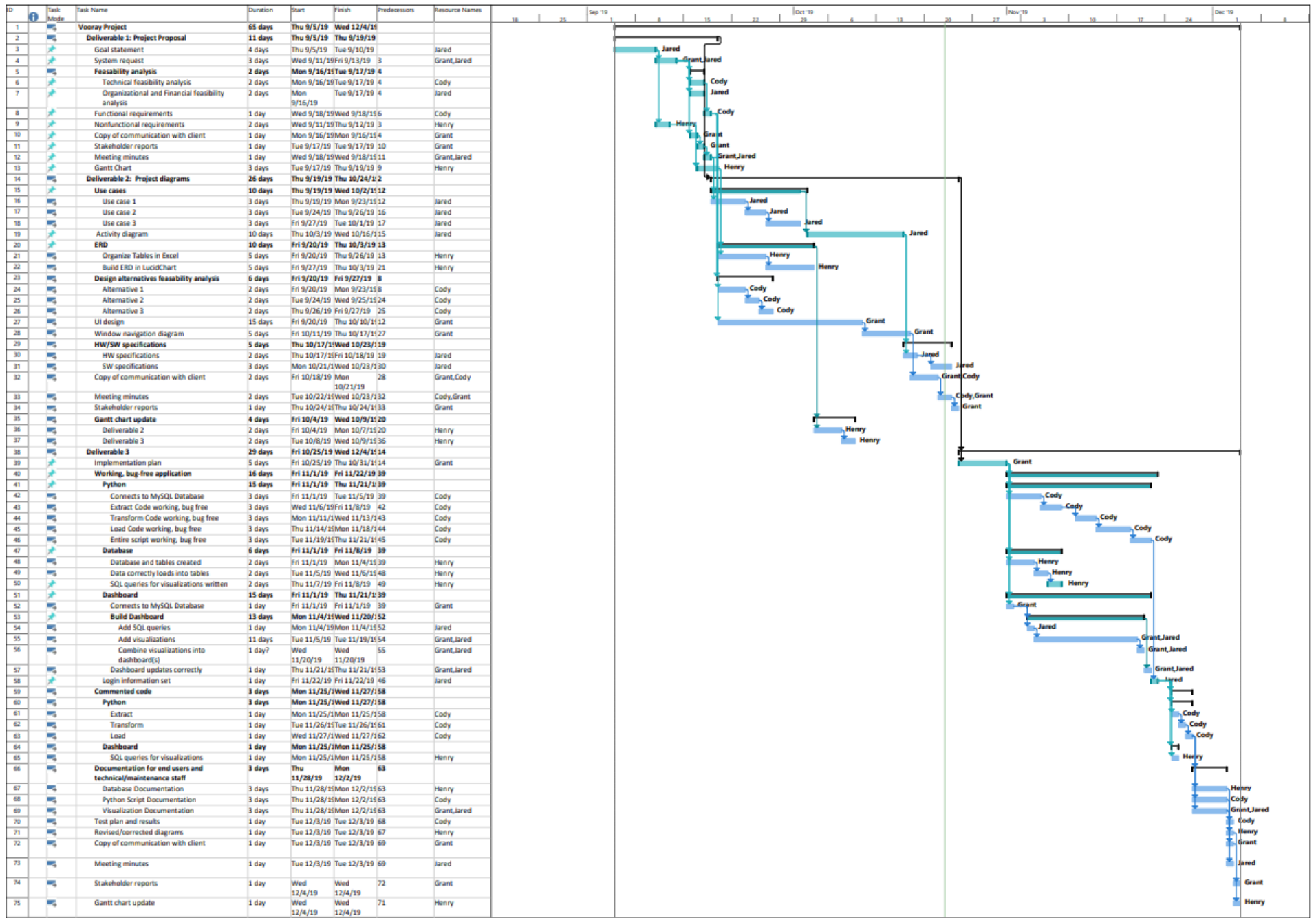
### ASSUMPTIONS

---

In order to better assess our progress, more subtasks have been added to Deliverables 2 and 3. This will enable our team to be more accountable when it comes to implementing our solution. One who is in charge of completing a task will know if they are making progress on it if they have completed the subtasks laid out in the Gantt Chart.

---

<sup>4</sup> The Gantt chart is subject to change depending on the length of task, skill set, and the needs of our client.





---

## DELIVERABLE 3

---

12/5/19



## IMPLEMENTATION PLAN

---

Our implementation plan with the client will involve a meeting with the client where we will do the following steps for setup:

- Download Python on the client's computer
- Download MySQL on the client's computer
- Download PowerBI on the client's computer
- Download necessary connectors on the client's computer
- Download script files on the client's computer

Then the training on the system will involve the following steps:

- Help the client set up a folder to keep the script files and where downloaded data files will be located
- Show the client how to activate the script using the selected files they want to use
- Show the client the database and how to access it if they want it to see the raw data
- Show the client the PowerBI visualization and how to navigate it
- Answer any application questions

## POTENTIAL ISSUES

---

- If the CSV file format used by Amazon changes, especially column names, then the script could cease to work and the code would need to be updated, we will explain this to the client



## TEST PLANS AND TEST RESULTS

Tester: Cody Crofoot			Date Conducted: 11/28		
Testing Objectives: Testing the creation of the database					
Event	Action	Value Entered	Expected Result	Actual Result	Result P/F
Connecting to MySQL	Update line 2 in create_db_c.py		Display prints: MySQLCursor: (Nothing executed yet)	Display prints: MySQLCursor: (Nothing executed yet)	P
Creating item table	Run create_db_c.py		Item table is created in MySQL Workbench	Item table is created in MySQL Workbench	P
Creating inventory level table	Run create_db_c.py		Inventory Level table is created in MySQL Workbench	Inventory Level table is created in MySQL Workbench	P
Creating Amz_price table	Run create_db_c.py		AMZ_Price table is created in MySQL Workbench	AMZ_Price table is created in MySQL Workbench	P
Creating Quantity Sold table	Run create_db_c.py		Quantity_sold table is created in MySQL Workbench	Quantity_sold table is created in MySQL Workbench	P



Tester: Cody Crofoot			Date Conducted: 12/2/2019		
Testing Objectives: Testing Inserting data into database					
Event	Action	Value Entered	Expected Result	Actual Result	Result P/F
Connecting to MySQL	Update line 26-30 in script_c.py		Display prints: MySQLCursor: (Nothing executed yet)	Display prints: MySQLCursor: (Nothing executed yet)	P
Pull inventory information from csv	Update inv variable (line 10) with path of desired csv	'Inventory_1127.csv'	Print(inv_df.head()) will display data from csv	Print(inv_df.head()) will display data from csv	P
Pull price and quantity information from csv	Update prc variable (line 10) with path of desired csv	'OrderReport_1127.csv'	Print(prc_df.head()) will display data from csv	Print(prc_df.head()) will display data from csv	P
Update the Item Table	Run script_c.py (lines 104-190)		MySQL workbench reflects data from the CSV	MySQL workbench reflects data from the CSV	P
Update the Inv_Level Table	Run script_c.py (lines 193-270)		MySQL workbench reflects data from the CSV	MySQL workbench reflects data from the CSV	P
Update the AMZ_Price Table	Run script_c.py (lines 272 – 355)		MySQL workbench reflects data from the CSV	MySQL workbench reflects data from the CSV	P
Update the quantity_sold table	Run script_c.py (lines 357-433)		MySQL workbench reflects data from the CSV	MySQL workbench reflects data from the CSV	P



Tester: Cody Crofoot			Date Conducted: 12/2		
Testing Objectives: Testing duplicate logic of the script					
Event	Action	Value Entered	Expected Result	Actual Result	Result P/F
Inserting into item table (skipping duplicates)	Run script_c.py (lines 104-190) twice		Information should not be duplicated	Information was not duplicated	P
Inserting into inv_level table (skipping duplicates)	Run script_c.py (lines 193-270) twice		Information should not be duplicated	Information was not duplicated	P
Inserting into Amz_price table (skipping duplicates)	Run script_c.py (lines 272 – 355) twice		Information should not be duplicated	Information was not duplicated	P
Inserting into Quantity_sold table (skipping duplicates)	Run script_c.py (lines 357-433) twice		Information should not be duplicated	Information was not duplicated	P



Tester: Grant Brinkerhoff			Date Conducted: 12/2		
Testing Objectives: Visualizations					
Event	Action	Value Entered	Expected Result	Actual Result	Result P/F
Connecting to MySQL Workbench	Use PowerBI's MySQL connector to establish database link.		Connection should be established to the database. Should allow for data pull.	As expected.	P
Pulls all data in visualization	Accept all database access.		Data should populate within PowerBI.	As expected.	P
Pulls updated data in a visualization	Push refresh on the home ribbon		Data connection should be made again and charts should be updated with new data.	As expected	P
Correctly filters according to SKU	None required, built into Visualization		Visualization should only have set SKUs.	As expected	P
Correctly displays amounts	None required, built into Visualization		Visualization should set SKU to date, inventory, and order amounts.	As expected	P





## DOCUMENTATION

### DOCUMENTATION FOR END USERS

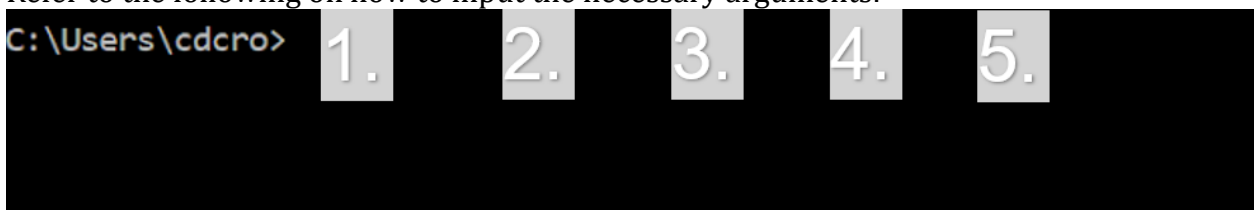
#### RUNNING THE SCRIPT:

*Please see the Technical/Maintenance Documentation for any questions on how to set up the script.*

1. Open up the command prompt on your computer.



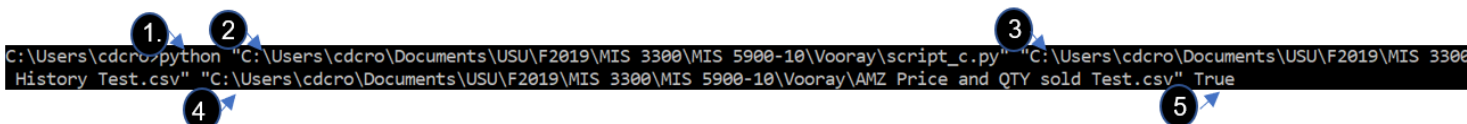
2. Refer to the following on how to input the necessary arguments:



1. Type "python" right here
2. Drag and drop the script after python
3. Drag and drop the CSV for inventory information
4. Drag and drop the CSV for the price and sales information
5. Type "True" (make sure it is capitalized)

Make sure there a space between each argument.

When you have done this, it should look similar to the following screenshot:

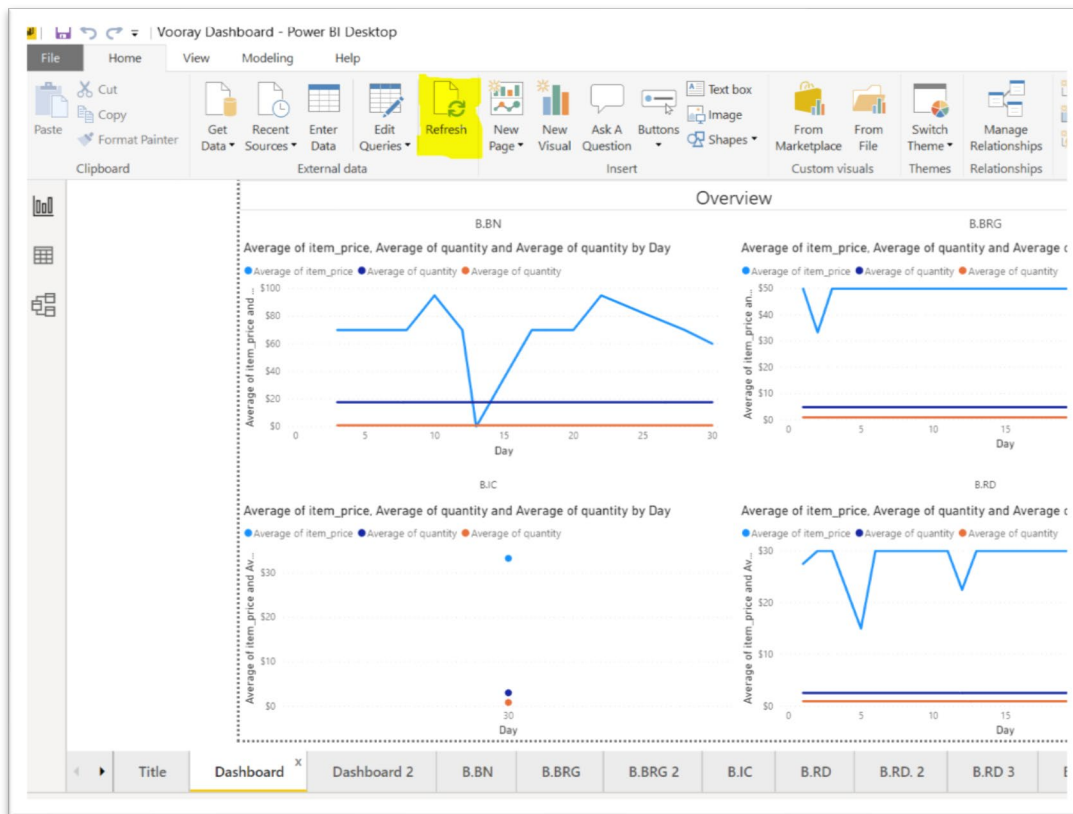


3. Hit Enter and let the program run. You can let this run in the background. It will say "COMPLETED" when it is finished.



## USING THE VISUALIZATIONS

1. Using PowerBI, and its visualizations are extremely simple. Once the data has been updated using the script, all that needs to be done on PowerBI is to click on the “Refresh” button in the Home Ribbon.



2. PowerBI should take a few seconds to gather the new data and then update visualizations.

3. From there, you should be able to sift through the dashboards and use the mouse to check individual data



## DOCUMENTATION FOR TECHNICAL/ MAINTENANCE

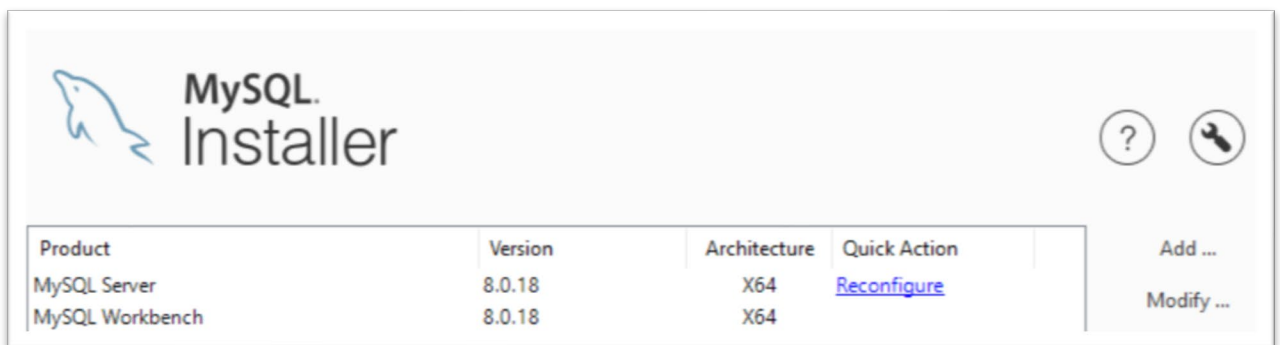
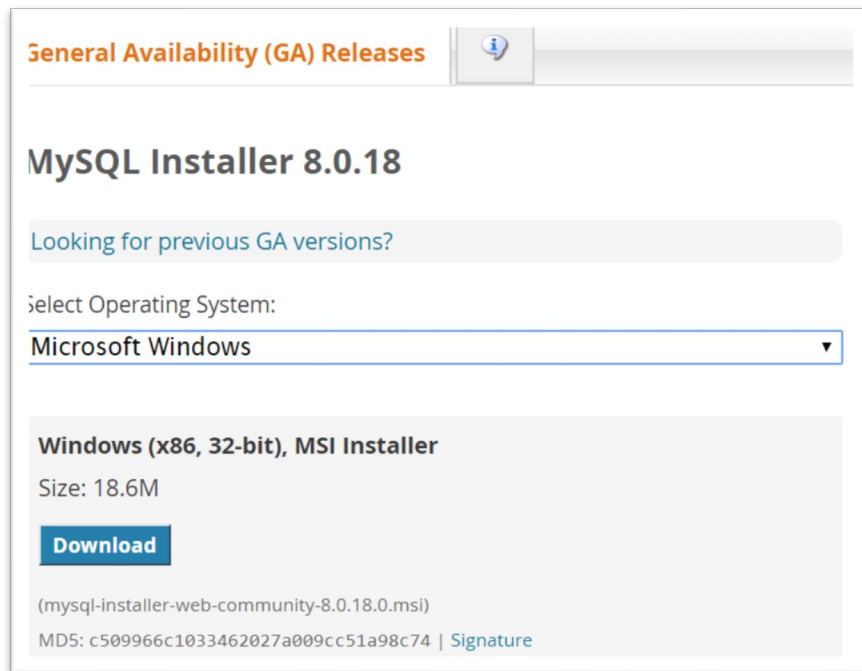
### SET UP:

#### Software:

In order for the program to function, you will need to download some initial programs that are necessary for it to work.

1. MySQL server and Workbench: These are the necessary programs in order to save the data in a database.

<https://dev.mysql.com/downloads/installer/>



**\*\*Use the Add button to install the two products displayed**

2. Python: You will need python installed on your computer in order to run the scripts. Download the most recent version of python from the following link:

<https://www.python.org/downloads/>



3. MySQL-connector-python: This is a python module used to connect to the database. It is a simple install with pip (which comes with python). Run the following from the command line: pip install MySQL-connector-python

---

### SETTING UP THE DATABASE:

---

Now that the necessary software is installed on your computer it is time to set up the database.

1. Log onto MySQL Workbench:  
There should be a MySQL Workbench app available on your computer. Open it up and you should see the following screen:

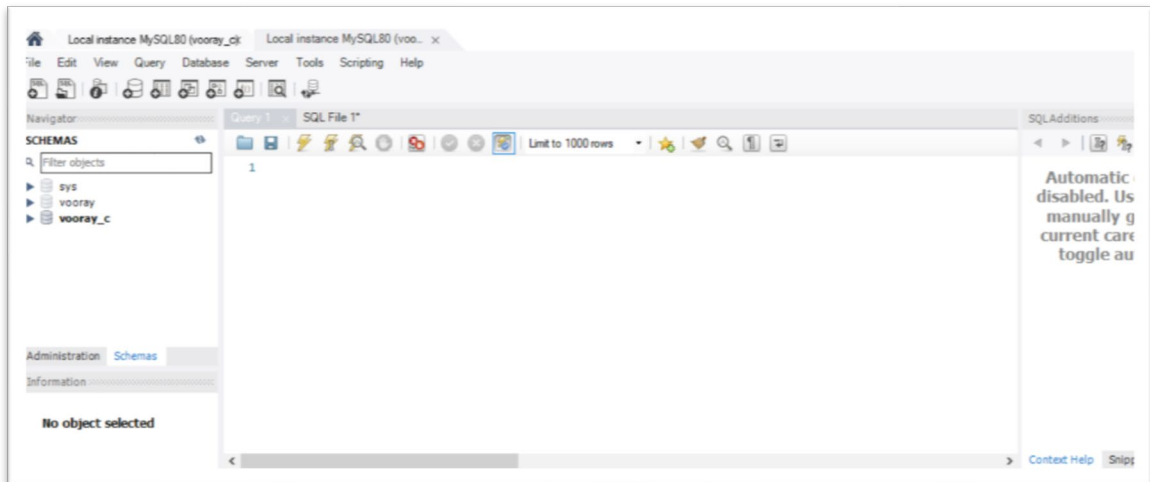


Now you should be able to log onto a local instance. The script uses the following credentials:

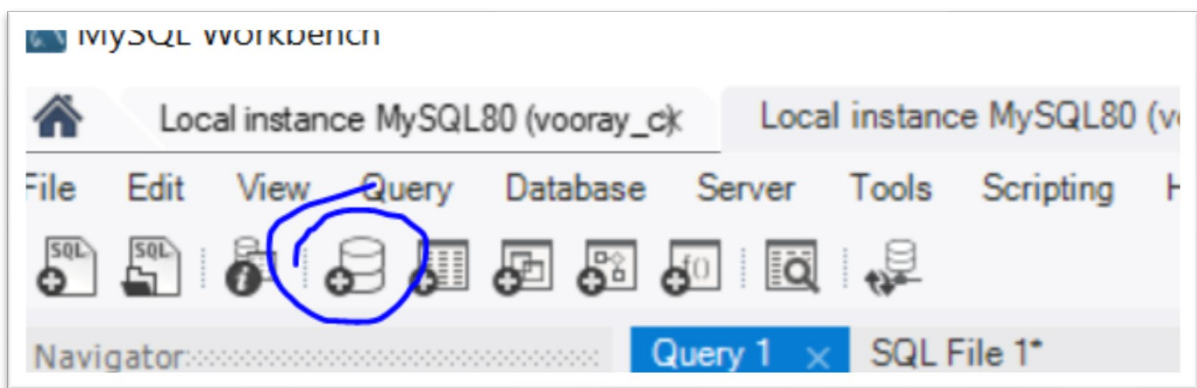
```
host="localhost",  
user="root",  
database="vooray_c",  
password="mis5900vooray"
```

These can be updated to whatever your preference is but make sure to update it in the script as well.

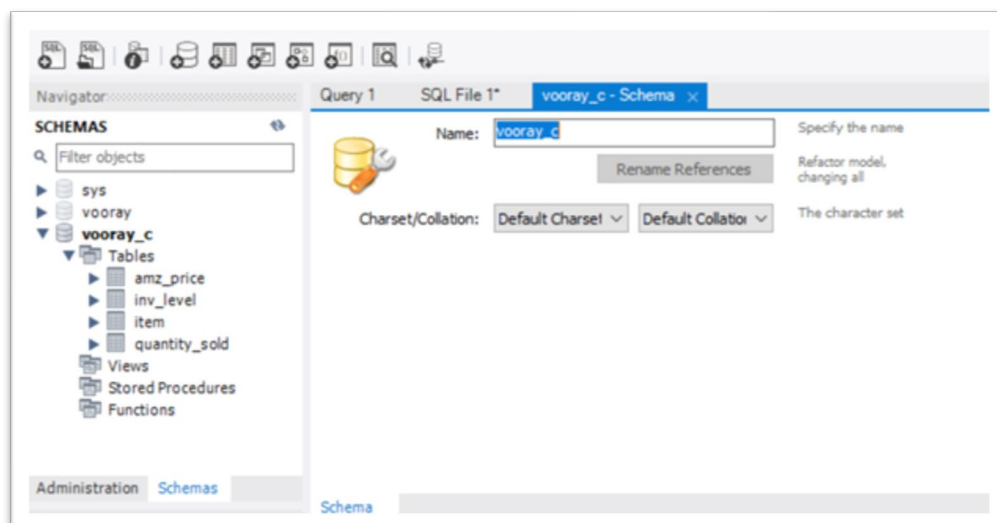
2. Create a new schema:  
Once you are logged into your local instance you should see the following:



In order to create a new schema, click the following button:

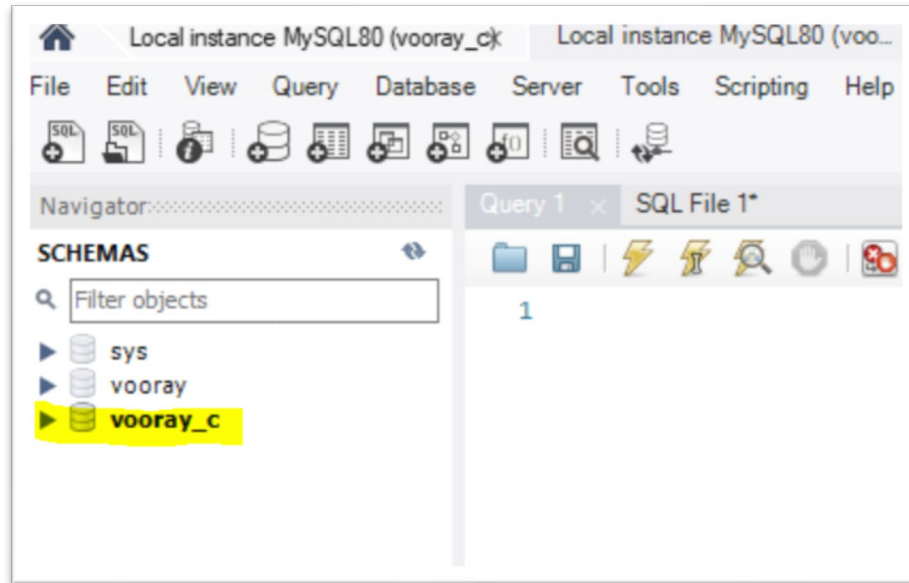


Name the schema: vooray\_c





Vooray\_c should now be available

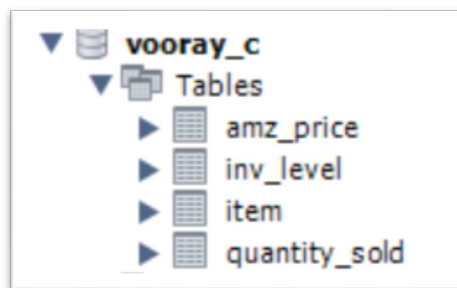


It should currently be empty and that is ok. The tables will not show up until after you run the create\_db\_c.py script

### 3. Create the tables

We have created a quick script called create\_db\_c.py that should do this for you. It is simple to run from the command line. Simply type python and space. Now drag and drop the file from whatever folder it is in on your computer.

It should end up looking something like this with the appropriate changes depending on where it is installed on your computer.



---

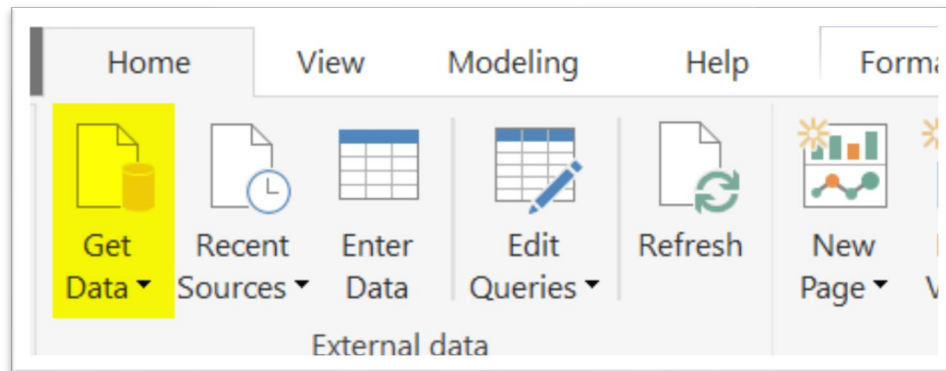
## CONNECTING TO POWER BI:

---

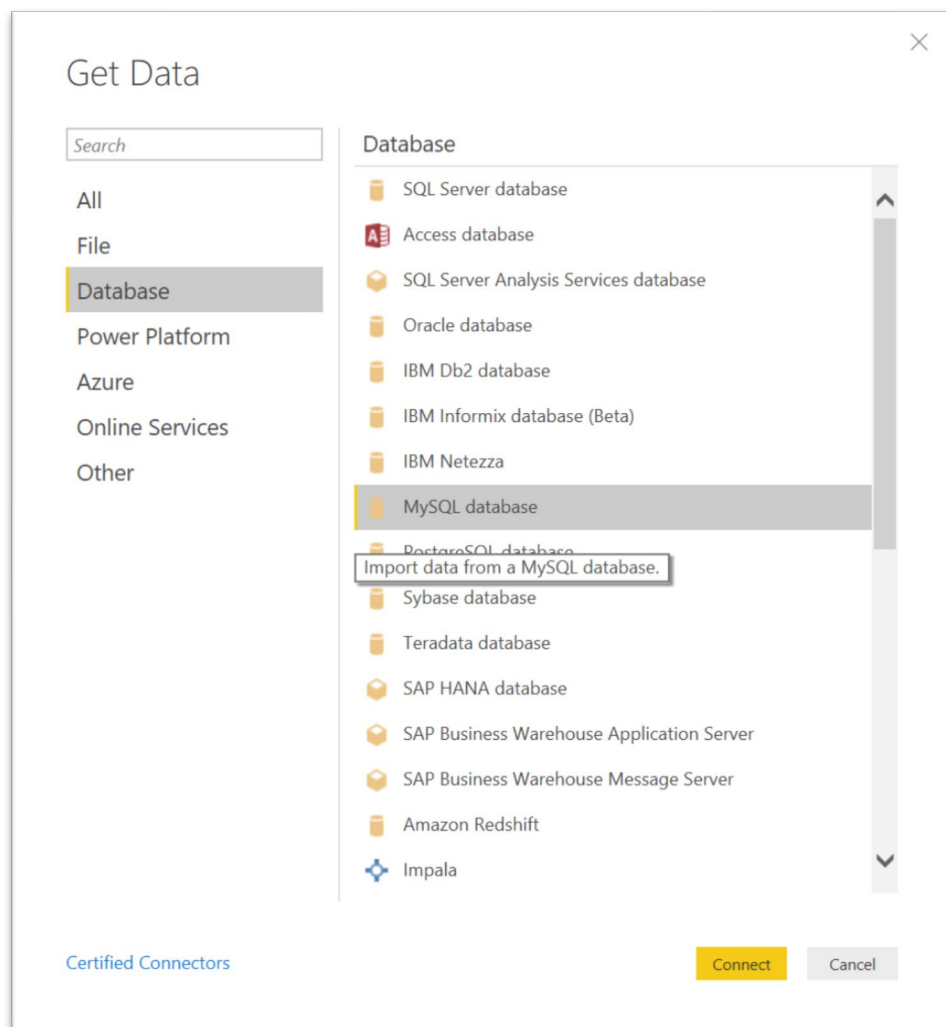
Once PowerBI is successfully downloaded and installed, a connection is needed between PowerBI and the local MySQL database.



1. Open the supplied visualization file. Error signs should be present as there is no data connection to populate the visualizations.
2. Click on the “Get Data” tab in the “Home” ribbon.



3. Once in the Get Data dialog box, select “Database” from the left side boxes and then select the “MySQL database” from the right-hand side selection box.





- The next box contains the needed information to make the connection to the MySQL database. Since MySQL is locally hosted, the Server is going to be “Localhost” and the database will be the name of the database set within the code, in this case, “vooray\_c”.

MySQL database

Server  
localhost

Database  
vooray\_c

Advanced options

OK Cancel

- Within the next box, select “Database” on the left-hand side. The User Name for the database is root and the password is “mis5900vooray”, the same as the MySQL Workbench.

MySQL database

localhost;vooray\_c

User name  
root

Password  
.....

Select which level to apply these settings to  
localhost

Back Connect Cancel

- From there, all that is needed is to select all of the tables and continue on. PowerBI should then start importing the data and applying it to the visualizations.





7. Because MySQL is locally hosted, using the provided PowerBI template will actually work since all of the information is the same on the connection ports.

---

#### CODE OVERVIEW:

---

---

#### CREATE\_DB\_C.PY OVERVIEW:

---

```
import mysql.connector
```

You need to have the MySQL-connector-python modules installed in order for the script to run properly.

```
conn = mysql.connector.connect(host="localhost", user="root", database="vooray",  
password="mis5900vooray")
```

Update the credentials if needed

The program has four functions:

```
create_item()  
create_inv_level()  
create_amz_price()  
create_quantity_sold()
```

Each function creates its respective table through SQL instructions.

---

#### SCRIPT\_C.PY OVERVIEW:

---

Lines 1-5:

```
import pandas as pd  
import numpy as np  
import mysql.connector  
import time  
import sys
```

These are the necessary python modules that need to be installed in order for the script to run.

Lines 7-26:

```
#####  
# VARIABLES #
```



```
#####
```

```
script = sys.argv[0]  
inv = sys.argv[1]  
prc = sys.argv[2]  
full_run = sys.argv[3]  
# inv = 'Inventory_1127.csv'  
# prc = 'OrderReport_1127.csv'  
# full_run = True
```

```
if full_run:  
    num = 500000 # The amount of records you want to run through, for a full run number is  
    arbitrarily high  
else:  
    num = 10
```

```
#####  
# END VARIABLES #  
#####
```

This is the variable section of the program. It is the only section that should have to be updated each time the code works. Currently, the code is taking 4 variables as arguments in the command line. These can also be specified in the code if needed (see the commented code in lines 15-17).

script: This is the name of the program.

Inv: This should be set to the path to the inventory report CSV

Prc: This should be set to the path to the price and quantity sold report CSV

Full\_run: set this to true if you want to run through every record in the CSV. This should usually be the case but you can set it to lower amounts (put false) if you want to do some quick testing.

Lines 50-84:

These set up the templates for the data to be put into the database. There are four empty templates modeled after the four tables in the database. If there are any modifications to the tables in the database (adding additional columns, etc.) this should be reflected in these templates.

Lines 86-104:

This section puts the data from the two CSVs into a data frame. We drop unnecessary columns at this point and reformat the date information.



Lines 113-199:

This section updates the item table. It looks through each unique SKU in both Data Frames and pulls the necessary information for them.

Lines 201-270:

This section updates the Inv\_Level Table. It loops through each unique date for each SKU and adds it to the Database.

Lines 272 - 352:

This section updates the AMZ\_Price table. It loops through each date and takes the associated price for that day.

Lines 354 - 422:

This section updates the Quantity Sold Table.

The code is commented throughout in each section to describe what is happening and is a good reference if there are issues.

#### **DROP\_DB\_C.PY OVERVIEW:**

---

**DO NOT RUN THIS UNLESS YOU ABSOLUTELY NEED TO.**

This file will delete all of the data in the database. It is useful if you feel like you no longer need the old data or there is an issue and you want to restart. Once you run this script there is no way to go back and restore the data that was lost without uploading it all over again.



## CREATION OF DATABASE

---

```
import mysql.connector
conn = mysql.connector.connect(host="localhost", user="root", database="vooray", password="mis5900vooray")

curs = conn.cursor()
print(curs)

def create_item():
    curs.execute(
        f"""
        CREATE TABLE `vooray_c`.`item` (
          `fnsku` VARCHAR(60) NULL,
          `sku` VARCHAR(45) NOT NULL,
          `product_name` VARCHAR(360) NULL,
          `asin` VARCHAR(60),
          PRIMARY KEY (`sku`));
        """)

def create_inv_level():
    curs.execute(
        f"""
        CREATE TABLE `vooray_c`.`inv_level` (
          `s_date` VARCHAR(60),
          `date` DATE NOT NULL,
          `quantity` INT NULL,
          `fulfillment_center_id` VARCHAR(20) NULL,
          `detailed_disposition` VARCHAR(45) NULL,
          `country` VARCHAR(10) NULL,
          `sku` VARCHAR(45) NULL);
        """)

def create_amz_price():
    curs.execute(
        f"""
        CREATE TABLE `vooray_c`.`amz_price` (
          `sku` VARCHAR(45) NULL,
          `p_date` VARCHAR(60),
          `date` DATE NOT NULL,
          `amazon_order_id` VARCHAR(45) NULL,
          `fulfillment_channel` VARCHAR(45) NULL,
          `sales_channel` VARCHAR(45) NULL,
          `currency` VARCHAR(10) NULL,
          `item_price` FLOAT NULL,
          `item_tax` FLOAT NULL,
          `shipping_price` FLOAT NULL,
          `shipping_tax` FLOAT NULL,
          INDEX `sku_idx` (`sku` ASC) VISIBLE,
          CONSTRAINT `sku`
            FOREIGN KEY (`sku`)
            REFERENCES `vooray_c`.`item` (`sku`)
            ON DELETE NO ACTION
        """)
```



```
ON UPDATE NO ACTION);  
")
```

```
def create_quantity_sold():  
    curs.execute(  
        f"  
        CREATE TABLE `vooray_c`.`quantity_sold` (  
            `sku` VARCHAR(45) NULL,  
            `p_date` VARCHAR(60),  
            `date` DATE NOT NULL,  
            `quantity` INT NULL,  
            `amazon_order_id` VARCHAR(45) NULL,  
            `fulfillment_channel` VARCHAR(45) NULL,  
            `sales_channel` VARCHAR(45) NULL,  
            INDEX `sku_idx` (`sku` ASC) VISIBLE,  
            CONSTRAINT ``  
            FOREIGN KEY (`sku`)  
            REFERENCES `vooray_c`.`item` (`sku`)  
            ON DELETE NO ACTION  
            ON UPDATE NO ACTION);  
        ")
```

```
def main():  
    create_item()  
    create_inv_level()  
    create_amz_price()  
    create_quantity_sold()
```

```
main()
```



## DROP DATABASE

---

```
import mysql.connector
conn = mysql.connector.connect(host="localhost", user="root", database="vooray",
password="mis5900vooray")

curs = conn.cursor()
print(curs)

def main():
    curs.execute(
        f"
        DROP TABLE `vooray_c`.`amz_price`, `vooray_c`.`inv_level`, `vooray_c`.`item`, `vooray_c`.`quantity_sold`;
        ")

main()
```



## DATABASE POPULATION SCRIPT (COMMENTED)

---

```
import pandas as pd
import numpy as np
import mysql.connector
import time
import sys

#####
# VARIABLES #
#####

script = sys.argv[0]
inv = sys.argv[1]
prc = sys.argv[2]
full_run = sys.argv[3]
# inv = 'Inventory_1127.csv'
# prc = 'OrderReport_1127.csv'
# full_run = True

if full_run:
    num = 5000000 # The amount of records you want to run through, for a full run number is arbitrarily
    high
else:
    num = 10

#####
# END VARIABLES #
#####

print("STARTING")
start = time.time()

# Connect to the database
conn = mysql.connector.connect(host="localhost",
                               user="root",
                               database="vooray_C",
                               password="mis5900vooray")

curs = conn.cursor()
print(curs)

#####
# CREATE TEMPLATE OF DATA FRAMES TO GO IN DATABASE #
#####

df_index = [0]

# item table
item_template = {'fnsku': np.nan,
```



```
'sku': np.nan,
'product_name': np.nan,
'asin': np.nan}
item = pd.DataFrame(item_template, df_index)

# inv_level table
inv_level_template = {'s_date': np.nan,
                      'date': np.nan,
                      'quantity': np.nan,
                      'fulfillment_center_id': np.nan,
                      'detailed_disposition': np.nan,
                      'country': np.nan,
                      'sku': np.nan}
inv_level = pd.DataFrame(inv_level_template, df_index)

# amz_price
amz_price_template = {'sku': np.nan,
                      'p_date': np.nan,
                      'date': np.nan,
                      'amazon_order_id': np.nan,
                      'fulfillment_channel': np.nan,
                      'sales_channel': np.nan,
                      'currency': np.nan,
                      'item_price': np.nan,
                      'item_tax': np.nan,
                      'shipping_price': np.nan,
                      'shipping_tax': np.nan}
amz_price = pd.DataFrame(amz_price_template, df_index)

# quantity sold
quantity_sold_template = {'sku': np.nan,
                          'date': np.nan,
                          'quantity': np.nan,
                          'amazon_order_id': np.nan,
                          'fulfillment_channel': np.nan,
                          'sales_channel': np.nan}
quantity_sold = pd.DataFrame(quantity_sold_template, df_index)

#####
# NEXT ACQUIRE THE DATA AND TRANSFORM IT #
#####

# get the data
inv_df = pd.read_csv(inv)
prc_df = pd.read_csv(prc)

# Remove white space
inv_df.columns = inv_df.columns.str.replace(' ', '')
prc_df.columns = prc_df.columns.str.replace(' ', '')
```





```
# print(list(prc_df))
prc_droplist = ['merchant-order-id', 'last-updated-date', 'order-status',
               'order-channel', 'url', 'ship-service-level', 'item-status', 'gift-wrap-price', 'gift-wrap-tax',
               'item-promotion-discount', 'ship-promotion-discount', 'ship-city', 'ship-state', 'ship-postal-code',
               'ship-country', 'promotion-ids']
prc_df = prc_df.drop(prc_droplist, axis=1)

# Fix date
inv_df['date'] = inv_df['snapshot-date'].str.split('T').str[0]
inv_df['date'] = inv_df['snapshot-date'].astype('datetime64[ns]')
prc_df['date'] = prc_df['purchase-date'].str.split('T').str[0]
prc_df['date'] = prc_df['purchase-date'].astype('datetime64[ns]')

#####
# STARTING WITH UPDATING THE ITEM TABLE #
#####

# This info is found in both inventory count and amazon, TODO: make sure they are consistent across
# print(len(prc_df['sku'].unique()))
# print(len(inv_df['sku'].unique()))
sku_list = []

# Look over SKUs on inventory df
skus = inv_df['sku'].unique()
i = 1
for sku in skus:
    # Subset Data by sku
    df = inv_df[inv_df.sku == sku]
    df2 = prc_df[prc_df.sku == sku]

    # SET THE VARIABLES TO GO INTO THE DATABASE TODO: (should i use "" or NULL?)
    fnsku = df['fnsku'].unique()[0] if len(df['fnsku'].unique()) == 1 else ""
    sku = sku
    product_name = df['product-name'].unique()[0].replace("'", "") if len(df['product-name'].unique()) == 1
    else ""
    asin = df2['asin'].unique()[0] if len(df2['asin'].unique()) == 1 else ""

    temp_dict = {'fnsku': fnsku,
                 'sku': sku,
                 'product_name': product_name,
                 'asin': asin}
    temp_index = [i]
    i += 1

    temp = pd.DataFrame(temp_dict, temp_index)
    frames = [item, temp]
    item = pd.concat(frames)
    sku_list.append(sku)

# Now go add any leftovers from the prc_df
```



```
skus = prc_df['sku'].unique()
for sku in skus:
    if not (sku in sku_list):
        # Subset Data by sku
        df = inv_df[inv_df.sku == sku]
        df2 = prc_df[prc_df.sku == sku]
        fnsku = df['fnsku'].unique()[0] if len(df['fnsku'].unique()) == 1 else ""
        sku = sku
        product_name = df['product-name'].unique()[0].replace("'", "") if len(df['product-name'].unique()) == 1
    else ""
    asin = df2['asin'].unique()[0] if len(df2['asin'].unique()) == 1 else ""

    temp_dict = {'fnsku': fnsku,
                  'sku': sku,
                  'product_name': product_name,
                  'asin': asin}
    temp_index = [i]
    i += 1

    temp = pd.DataFrame(temp_dict, temp_index)
    frames = [item, temp]
    item = pd.concat(frames)

# Update the Database #
item.drop(item.index[1:], inplace=True)

for index, row in item.iterrows():

    # Test to see if this piece of data is already in the database
    curs.execute(
        f"""
        SELECT sku
        FROM `vooray_c`.`item`
        WHERE sku = "{row['sku']}"
        """
    )
    results = curs.fetchall()

    # None of these should return more than one thing but just in case
    if (len(results) > 1):
        print(f"ERROR AT {row['sku']}")

    # If len is 0 that means it is not in the database
    if len(results) == 0:
        curs.execute(
            f"""
            INSERT INTO `vooray_c`.`item` (fnsku, sku, product_name, asin)
            VALUES ("{row['fnsku']}", "{row['sku']}", "{row['product_name']}", "{row['asin']}");
            """
        )
    conn.commit()
```



```
#####  
# Updating the Inv_Level Table #  
#####  
  
snapshots = inv_df['snapshot-date'].unique()  
skus = inv_df['sku'].unique()  
i = 1  
for snapshot in snapshots:  
    filter1 = inv_df['snapshot-date'] == snapshot  
    df = inv_df[filter1]  
  
    for index, row in df.iterrows():  
        s_date = row['snapshot-date']  
        date = row['date']  
        quantity = row['quantity']  
        fulfillment_center_id = row['fulfillment-center-id']  
        detailed_disposition = row['detailed-disposition']  
        country = row['country']  
        sku = row['sku']  
  
        temp_dict = {'s_date': s_date,  
                    'date': date,  
                    'quantity': quantity,  
                    'fulfillment_center_id': fulfillment_center_id,  
                    'detailed_disposition': detailed_disposition,  
                    'country': country,  
                    'sku': sku}  
        temp_index = [i]  
        i += 1  
        if i % 1000 == 0:  
            print("INV: ", i)  
  
        temp = pd.DataFrame(temp_dict, temp_index)  
        frames = [inv_level, temp]  
        inv_level = pd.concat(frames)  
        if i == num:  
            # print(inv_level)  
            break  
    if i == num:  
        break  
  
#update the database  
inv_level.drop(inv_level.index[:1], inplace=True)  
  
# print(inv_level.head())  
# print(list(inv_level))  
for index, row in inv_level.iterrows():  
    # Test to see if this piece of data is already in the database
```



```
curs.execute(
    f"""
    SELECT sku, date, fulfillment_center_id
    FROM `vooray_c`.`inv_level`
    WHERE sku = "{row['sku']}" AND fulfillment_center_id = "{row['fulfillment_center_id']}" AND
    DATE(date) = DATE("{row['date']}")
    """
)
results = curs.fetchall()
# None of these should return more than one thing but just in case
if (len(results) > 1):
    print(f"DUPLICATE AT {row}")

# If len is 0 that means it is not in the database
if len(results) == 0:
    curs.execute(
        f"""
        INSERT INTO `vooray_c`.`inv_level` ( s_date, date, quantity, fulfillment_center_id, detailed_disposition,
country, sku)
        VALUES ( "{row['s_date']}", "{row['date']}", "{row['quantity']}", "{row['fulfillment_center_id']}",
"{row['detailed_disposition']}", "{row['country']}", "{row['sku']}");
        """
    )
    conn.commit()

#####
# Updating the AMZ_Price #
#####
print("----STARTING AMZ_PRICE-----")
dates = prc_df['purchase-date'].unique()
i = 1

for date in dates:
    filter1 = prc_df['purchase-date'] == date
    df = prc_df[filter1]
    for index, row in df.iterrows():
        sku = row['sku']
        p_date = row['purchase-date']
        date = row['date']
        amazon_order_id = row['amazon-order-id']
        fulfillment_channel = row['fulfillment-channel']
        sales_channel = row['sales-channel']
        currency = row['currency']
        item_price = row['item-price'] if not np.isnan(row['item-price']) else 0
        item_tax = row['item-tax'] if not np.isnan(row['item-tax']) else 0
        shipping_price = row['shipping-price'] if not np.isnan(row['shipping-price']) else 0
        shipping_tax = row['shipping-tax'] if not np.isnan(row['shipping-tax']) else 0

        temp_dict = {'sku': sku,
                     'p_date': p_date,
                     'date': date,
```



```
'amazon_order_id': amazon_order_id,
'fulfillment_channel': fulfillment_channel,
'sales_channel': sales_channel,
'currency': currency,
'item_price': item_price,
'item_tax': item_tax,
'shipping_price': shipping_price,
'shipping_tax': shipping_tax}
temp_index = [i]
i += 1
if i % 1000 == 0:
    print("PRICE: ", i)

temp = pd.DataFrame(temp_dict, temp_index)
frames = [amz_price, temp]
amz_price = pd.concat(frames)
if i == num:
    break
if i == num:
    break

# Update the Database #
amz_price.drop(amz_price.index[:1], inplace=True)

# print(amz_price.head())
# print(list(amz_price))
# print(amz_price)
# print(amz_price['amazon_order_id'])

for index, row in amz_price.iterrows():

    # Test to see if this piece of data is already in the database
    curs.execute(
        f"""
        SELECT sku
        FROM `vooray_c`.`amz_price`
        WHERE sku = "{row['sku']}" AND amazon_order_id = "{row['amazon_order_id']}" AND DATE(date) =
        DATE("{row['date']}")
        """
    )
    results = curs.fetchall()

    # None of these should return more than one thing but just in case
    if (len(results) > 1):
        print(f"DUPLICATE AT {row['sku']}")

    # If len is 0 that means it is not in the database
    if len(results) == 0:
        curs.execute(
            f"""
```



```
INSERT INTO `vooray_c`.`amz_price`(sku, p_date, date, amazon_order_id, fulfillment_channel,
sales_channel, currency, item_price, item_tax, shipping_price, shipping_tax)
VALUES ("{row['sku']}", "{row['p_date']}", "{row['date']}", "{row['amazon_order_id']}",
"{row['fulfillment_channel']}", "{row['sales_channel']}", "{row['currency']}", "{row['item_price']}",
"{row['item_tax']}", "{row['shipping_price']}", "{row['shipping_tax']}");
""
```

```
)
conn.commit()
```

```
#####
# Updating the quantity_sold Table #
#####
```

```
snapshots = prc_df['purchase-date'].unique()
skus = prc_df['sku'].unique()
# print(len(snapshots))
i = 1
for snapshot in snapshots:
    filter1 = prc_df['purchase-date'] == snapshot
    df = prc_df[filter1]

    for index, row in df.iterrows():
        sku = row['sku']
        p_date = row['purchase-date']
        date = row['date']
        quantity = row['quantity']
        amazon_order_id = row['amazon-order-id']
        fulfillment_channel = row['fulfillment-channel']
        sales_channel = row['sales-channel']

        temp_dict = {'sku': sku,
                     'p_date': p_date,
                     'date': date,
                     'quantity': quantity,
                     'amazon_order_id': amazon_order_id,
                     'fulfillment_channel': fulfillment_channel,
                     'sales_channel': sales_channel}
        temp_index = [i]
        i += 1
        if i % 1000 == 0:
            print("QUANTITY: ", i)

        temp = pd.DataFrame(temp_dict, temp_index)
        frames = [quantity_sold, temp]
        quantity_sold = pd.concat(frames, sort=True)
        if i == num:
            # print(quantity_sold)
            break
    if i == num:
        break
```



```
# update the database
quantity_sold.drop(quantity_sold.index[:1], inplace=True)

# print(quantity_sold.head)
for index, row in quantity_sold.iterrows():
    # Test to see if this piece of data is already in the database
    curs.execute(
        f"""
        SELECT sku, date, quantity, amazon_order_id
        FROM `vooray_c`.`quantity_sold`
        WHERE sku = "{row['sku']}" AND quantity = "{row['quantity']}" AND DATE(date) =
        DATE("{row['date']}") AND amazon_order_id = "{row['amazon_order_id']}"
        """
    )
    results = curs.fetchall()
    # None of these should return more than one thing but just in case
    if (len(results) > 1):
        print(f"DUPLICATE AT {row}")

    # If len is 0 that means it is not in the database
    if len(results) == 0:
        curs.execute(
            f"""
            INSERT INTO `vooray_c`.`quantity_sold` ( sku, p_date, date, quantity, amazon_order_id,
            fulfillment_channel, sales_channel)
            VALUES ( "{row['sku']}", "{row['p_date']}", "{row['date']}", "{row['quantity']}",
            "{row['amazon_order_id']}", "{row['fulfillment_channel']}", "{row['sales_channel']}");
            """
        )
        conn.commit()

end = time.time()

print("SUCCESS!")
print(end - start)
```



## CORRECTED DIAGRAMS

### UPDATED USE CASES

<b>Use Case Name:</b> Download Data	<b>ID:</b> 001	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The user goes to amazon and downloads the sales they would like cleansed and uploaded to the database		
<b>Trigger:</b> User wants to initiate the process  <b>Type:</b> External		
Relationships: Association: Include: Extend: Generalization:		
<b>Normal Flow of Events:</b>  1. Login to Amazon Account 2. Go to order history reports in Your Account 3. Select the report type from the drop-down menu, then fill in the start date, end date, and report name. Click Request Report. 4. When the report is complete, Amazon will send/receive an e-mail notification. To retrieve the report, visit Order History Reports and click Download. 5. Move data CSV file to Project_Folder		
SubFlows:		
Alternate/ Exceptional Flows:		





<b>Use Case Name:</b> Clean Data	<b>ID:</b> 002	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The user opens up the command prompt and runs our script with desired files to initiate application		
<b>Trigger:</b> User runs the application from the command prompt Type: External		
<b>Relationships:</b> Association: Include: Extend: Load Data Generalization:		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"><li>1. The user initiates our application from the command prompt</li><li>2. Dates are changed to date format from dateTime format</li><li>3. Trim column names</li><li>4. Drop 16 unnecessary/unused columns</li><li>5. Saves cleansed file</li></ol>		
Sub Flows:		
Alternate/Exceptional Flows:		



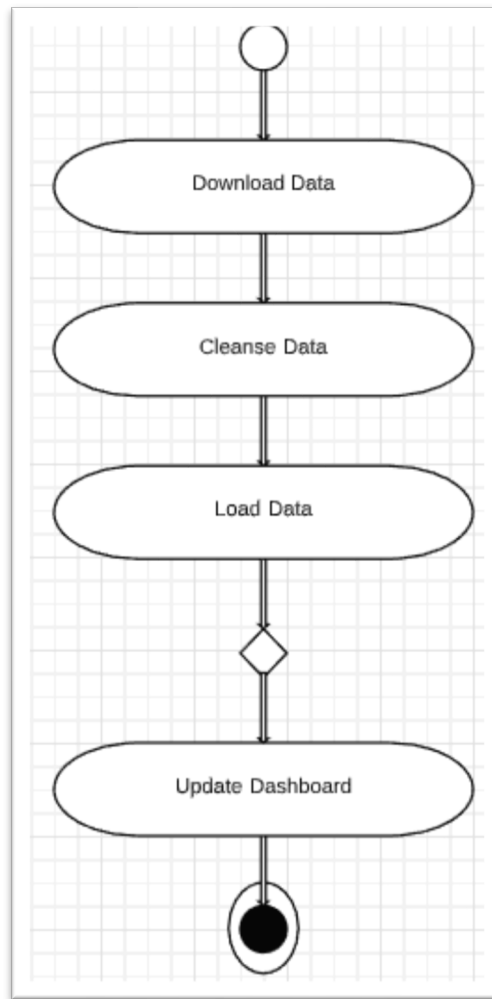
<b>Use Case Name:</b> Load Data	<b>ID:</b> 003	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> Data in CSV file that was already cleansed is uploaded to the database from Python Pandas Dataframe		
<b>Trigger:</b> Data extraction/cleansing is complete <b>Type:</b> Internal		
<b>Relationships:</b> Association: Include: Load Data Extend: Generalization:		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"><li>1. The script logs into the MySQL database on the computer as it already has stored login info</li><li>2. The script runs SQL code to create tables if they do not currently exist (should only happen once unless database data is lost)</li><li>3. The script runs SQL code to pull data from the Pandas Dataframe file and inserts it into the database</li></ol>		
SubFlows:		
Alternate/ Exceptional Flows		



<b>Use Case Name:</b> Update Dashboard	<b>ID:</b> 004	<b>Importance Level:</b> High
<b>Primary Actor:</b> Director of Finance	<b>Use Case Type:</b> Detail, Real	
<b>Stakeholders and Interests:</b> Director of Finance		
<b>Brief Description:</b> The user refreshes the Power BI		
<b>Trigger:</b> User selects the refresh <b>Type:</b> External		
<b>Relationships:</b> Association: Include: Extend: Generalization:		
<b>Normal Flow of Events:</b> 1. The user selects the refresh button on the top of the Power BI dashboard which updates to current database as they are linked		
SubFlows:		
Alternate/ Exceptional Flows:		



## UPDATED ACTIVITY DIAGRAM



## UPDATED WINDOWS NAVIGATION



```
C:\Users\cdcro>python "C:\Users\cdcro\Documents\USU\F2019\MIS_3300\MIS_5900-10\Vooray\script_c.py" "C:\Users\cdcro\Documents\USU\F2019\MIS_3300\MIS_5900-10\Vooray\Inventory History Test.csv" "C:\Users\cdcro\Documents\USU\F2019\MIS_3300\MIS_5900-10\Vooray\AMZ Price and QTY sold Test.csv" True
```

1. Type "python" right here
2. Drag and drop the script after python
3. Drag and drop the CSV for inventory information
4. Drag and drop the CSV for the price and sales information
5. Type "True" (make sure it is capitalized)



## COMMUNICATION WITH CLIENT

---

### TRANSCRIPT OF MEETINGS

---

**Time:** 9:00 AM

**Date:** November 20, 2019

**Location:** Remote Phone Call

**In Attendance:** Cody Crofoot

1. Talked about some last-minute understanding about the columns and what Brad is looking for in the visualizations. Brad provided a list of unnecessary columns and potential filters. We will keep only US and SELLABLE GOODS in the final visualization.
2. Talked about the final presentation and times that would be good for Brad.
3. Talked about potential user interfaces and what would be easiest.

### TEXT MESSAGES

---

---

10/19/19

---

**Cody:** Hey Brad, are you available to meet or for a call tomorrow morning about the project? I'm working it out and just wanted to walk through how you set up the Amazon daily sales and inventory analysis excel

**Brad:** Hey Cody, yeah, I can. It just depends on the time. When were you thinking?

**Cody:** I'm available anytime in the morning before 2:30

**Brad:** Does 9 am work?

**Cody:** Works for me, let's plan on it

**Brad:** Sounds good!

---

11/19/19

---

**Grant:** Hey Brad, are you free on the night of December 9<sup>th</sup> to listen to our Presentation for Vooray? They'd like you there for 30 minutes sometime between 6-9 PM if possible.

**Brad:** Hey Grant! Yeah, I can make that work. I'll put it into my calendar.



---

11/19/19

---

**Grant:** Hey Brad, are you free on the night of December 5<sup>th</sup> to listen to our Presentation for Vooray? They'd like you there for 30 minutes sometime between 6-9 PM if possible.

**Brad:** Hey Grant! Yeah, I can make that work. I'll put it into my calendar.

---

11/29/19

---

**Grant:** Hey, so our presentation is from about 6:20 – 6:50 PM. Can you make that?

**Brad:** Yeah, I can make that. Where is it at?

**Grant:** Huntsman Hall 124

**Brad:** Okay, yeah, I can make that.

---

#### OTHER CLIENT COMMUNICATION<sup>5</sup>

---

---

<sup>5</sup> No other client communication was needed for this deliverable.



## MEETING MINUTES

---

10/24/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, Jared, and Henry

### **Minutes:**

Presented the deliverable and then started talking about how we were going to make sure that we understand the requirements for the last deliverable and what we wanted to change.

### **Action Items**

Cody: Start looking at any design changes that we would like to make to the D3 memo and slide deck.

Grant: Start getting the D3 Memo formatted with the required sections that are needed. Set up a list of items that we are going to need.

Henry: Start looking at getting the MySQL database set up. Just make sure that we have all of the tables set up for the information dump.

Jared: Start making sure that we have meeting notes ready for the next memo.

10/28/19

---

**Time:** 11:00 am  
**Location:** Remote Call via Google Hangouts  
**In attendance:** Grant, Cody, Jared, and Henry

### **Minutes:**

We went over the action items from the last week. Henry is creating the tables and Grant is working on the

### **Action Items**

Cody: Start looking at any design changes that we would like to make to the D3 memo and slide deck. Look at helping Henry with the MySQL.

Grant: Start getting the D3 Memo formatted with the required sections that are needed. Set up a list of items that we are going to need.

Henry: Start looking at getting the MySQL database set up. Just make sure that we have all of the tables set up for the information dump.

Jared: Update the memo with the new minutes



10/31/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

We went over the feedback that we got from the last memo. We started to plan the changes that are going to be needed to make the next version of the memo even better. Henry has finished the MySQL table and Grant has finished updating the memo with the new required information.

**Action Items**

Cody: Look at trying to get everything into the database through a python script.  
Grant: Run the database by Johnson to make sure that it's set up in a way for PowerBI  
Henry: Respond to anything that Grant finds with Johnson  
Jared: Start looking at test cases for the documentation and PowerBI models.

11/04/19

---

**Time:** 11:00 AM  
**Location:** Remote call via Google Hangouts  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

Discussed study guides for the final exam. We reviewed the study guides and helped create study strategies.

**Action Items**

Cody: Focus on the final exam.  
Grant: Focus on the final exam.  
Henry: Focus on the final exam.  
Jared: Focus on the final exam.

11/07/19

---

**Time:** 5:15 pm  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**





We discussed the design of the database, the attributes from the given excel tables that we thought that we wanted to keep, and any other necessary data cleansing that was going to need to be done to make sure that our Use Cases were accurate.

### **Action Items**

Cody: Determine the coding platform and how the script will be run by the user.

Grant: Communicate with Brad to ensure that the necessary data is going to be stored in different segments of the database.

Henry: Begin working on creating a MySQL for the database and its necessary design.

Jared: Research and learn some python in preparation for programming.

11/11/19

---

**Time:** 11:00 AM

**Location:** Remote Call via Google Hangouts

**In attendance:** Grant, Cody, Jared, and Henry

### **Minutes:**

Reviewed the necessary documentation and tests required for D3, made plans to achieve. Begin making plans for paired programming for script, database work, documentation, and visualization.

### **Action Items**

Cody: Determine coding platform and how the script will be run by the user

Grant: Continue to research in connecting MySQL to PowerBI.

Henry: Continue working on SQL code for MySQL.

Jared: Continue to study python for paired programming.

11/14/19

---

**Time:** 5:15 PM

**Location:** Huntsman Hall 124

**In attendance:** Grant, Cody, Jared, and Henry

### **Minutes:**

Worked on the script and created a test database, began troubleshooting, importing, and cleaning of the data.

### **Action Items**

Cody: Get the python to MySQL connection to work for data insertion.

Grant: Continue research in connecting MySQL to PowerBI and continue the setup of visualizations.



Henry: Start making tweaks on SQL code with the python script.

Jared: Work on the data cleansing and checking for duplicates within the database insertion code.

11/18/19

---

**Time:** 11:00 AM  
**Location:** Remote Call via Google Hangouts  
**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

Discussed and make plans for completion of base program before the bug hunt, went over the current status of the project and how to get to completion. We went over changes in database design to make code more efficient for the clients.

**Action Items**

Cody: Meet on Wednesday to finish the script for the bug hunt.

Grant: Work on documentation.

Henry: Work on documentation.

Jared: Meet on Wednesday to finish the script for the bug hunt.

11/20/19

---

**Time:** 10:00 AM  
**Location:** Huntsman Hall  
**In attendance:** Cody & Jared

**Minutes:**

Finished the script that gets data from CSV into the MySQL database without errors or duplicates.

**Action Items**

Cody: Show up for the bug hunt.

Jared: Show up for the bug hunt.

11/21/19

---

**Time:** 5:15 PM  
**Location:** Huntsman Hall 124  
**In attendance:** Grant, Cody, Jared, and Henry



**Minutes:**

Participated in the bug hunt, reviewed feedback from bug hunt for project improvement.

**Action Items**

Cody: Work on getting MySQL to PowerBI connection working and updating.

Grant: Work on getting MySQL to PowerBI connection working and updating.

Henry: Work on documentation and memo

Jared: Work on documentation and memo.

11/25/19

---

**Time:** 7:15 PM

**Location:** Remote Call via Google Hangouts

**In attendance:** Grant, Cody, Jared, Henry, and Kathy

**Minutes:**

Meeting with Kathy to review the status of the project and how we will achieve completion by December 5<sup>th</sup>.

**Action Items**

Cody: Enjoy Thanksgiving!

Grant: Enjoy Thanksgiving!

Henry: Enjoy Thanksgiving!

Jared: Enjoy Thanksgiving!

Kathy: Enjoy Thanksgiving!

12/2/19

---

**Time:** 7:00 PM

**Location:** HH 136E

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

Finalize changes in the script and PowerBI. Worked on the documentation and the memo.

**Action Items**

Cody: Meet on Wednesday to finalize documentation and presentation.

Grant: Meet on Wednesday to finalize documentation and presentation.

Henry: Meet on Wednesday to finalize documentation and presentation.



Jared: Meet on Wednesday to finalize documentation and presentation.

12/4/19

---

**Time:** 7:00 PM

**Location:** HH 136E

**In attendance:** Grant, Cody, Jared, and Henry

**Minutes:**

Finalize the project documentation, memo, presentation, and created the presentation video.

**Action Items**

Cody: Show up to present tomorrow.

Grant: Format memo. Show up to present tomorrow.

Henry: Show up to present tomorrow.

Jared: Show up to present tomorrow.



## STAKEHOLDER REPORTS

---

Date: December 5, 2019

To: Brad Rigby, Vooray Director of Finance

From: King of Lions Design Team

Subject: Project Completion

Dear Mr. Rigby,

We are excited to reach out to you as we start to put together the finishing touches on the Vooray Project. We have worked hard to make sure that we are able to get you a working project that will help you to make better and more informed decisions in the future. We have gone through the development and testing processes, and are just finishing up the documentation.

As part of this, we have put together many useful sources of documentation to help with the implementation, testing, and use of the program. We have made sure that we have tested the system to get the best possible product for you and your team.

The program's function is easy to use as it is able to be used through your computer's command line or through an IDE such as PyCharm, Visual Studios, etc. Once done with that, your PowerBI dashboard will only need one button pressed to update all charts. We have done our best to optimize the program for speed cutting initial test runs by more than 1/3.

Brad, we hope to provide you with everything that you need to make this a good program for your company. We will be happy to help set this program up for you and to help you create any needed visualization in the future.

Please let us know if there is anything that we can do for you.

Sincerely,

The King of Lions Consulting Team



## GANTT CHART

---

