# Handover document

DECEMBER 3, 2022

**Better Business Bureau**

**Authored by: Data Divas (Medhanit Asrat Bekele, Mohammed Ahnaf Khalil, Wengel Tsegaselassie, Wen Sun)**

# TABLE OF CONTENTS

# 1. Introduction

BBB is a data company. It is imperative that the data in BBB's system is trustworthy and accurate. Unfortunately, collections of data (especially those as large and changeable as those managed by the BBB) can get faulty and unreliable. People visiting the BBB's web pages are looking for information on businesses. If that information is not accurate and informational, BBB loses trust in the marketplace. This project is about finding and implementing effective and powerful methods to improve data quality at the BBB, and thus improve the user experience for BBB consumers. In the past this kind of data management has often been done by hand. With our help, BBB hopes to convert that lengthy process to a more automatic and efficient process.

# 2. Getting started

## 2.1. Access to database

The first step in getting started with our project was to analyze the data in BBB's database. Our initial thought was to pock around the database; however, there were a few concerns about the permanent changes that it would cause. After discussing possible options with our client, we decided to export the data into a csv file, which makes it easier to manipulate the data without making permanent changes to the database. Our exported CSV file contains more than 150,000 businesses along with their necessary information. Since that file is too large, we decided to take a tranche of the data in order to test our code. To get the tranche of the data, we used a python package called pandas. Our tranche consisted of 100 rows from the table. We chose 100 rows because it was the ideal size to cover the majority of test cases without impeding execution. After getting the tranche, we planned to write a script that reads each URL from the tranche and analyzes the problem.

## 2.2. Skills learned

After speaking with our client, they expressed that they wanted us to use python for the script, so we did a few exercises to brush up our memory on python.

We also investigated a few helpful python packages that would aid us in data analysis as well as correcting data.

## 2.3. Documents developed

There were a couple of documents we developed to get started with this project.

1. **Timeline and Milestones** – is a document that uses milestones to divide project work into phases for easy monitoring. This document includes a list of critical dates and actions included in the project.
2. **Requirements Analysis Document -** document that includes full description of all types of data and databases to be included in the project, together with an analysis of all (or most, as possible) of the known problems in the data, together with typical downstream consequences (business and technical) associated with each type of problem.
3. **Test plan document** - document used to evaluate the code to determine whether it does what we expect it to do. Our test plan documents the testing strategies that will be used to ensure testing activities are effectively implemented within the Data Divas team.
4. **Development plan-** document that describes the strategy, plan and blueprint for the Project including any key milestone dates under the Project Development Documents and the estimated date of Practical Completion.

# 3. Analyzing data

After interviewing our client contacts, we learned that the problems with the URLs in the database are mostly as follows:

- Syntax error
- Redirect of business website URL
- Page not found
- Site cannot be reached

## 3.1. Packages used for data analysis

- Csv- to read the .csv files
- Re- regular expressions to check if a URL matches a pattern
- Urllib.parse- to process URLs, and to convert between URLs and platform-specific filenames
- Requests- to send HTTP requests using a Python code to get the status code of the URLs

- ThreadPoolExecutorPlus - to send a thread of HTTP requests
- Intertools - repeat the elements of the URLs in the list and the header to perform threading

## 3.2. Evaluating each URL for possible syntax problems

We evaluated the syntax for each URL from our tranche by using regex. We checked if the given URLs matched the expected pattern for a common URL. The most common error we observed was unnecessary special characters at the end of each URL.

## 3.3. Identifying the status code associated with each URL

The second step in the analysis of data was to write a code that reads each URL from our tranche and evaluates the status code that is associated with each URL. We used a python package called requests that given the URL, it will return the status code. We recorded each URL with their observed errors in a different CSV file for easier testing and observation.

## 3.4. Challenges faced with data analysis

### 3.4.1. Syntax problem

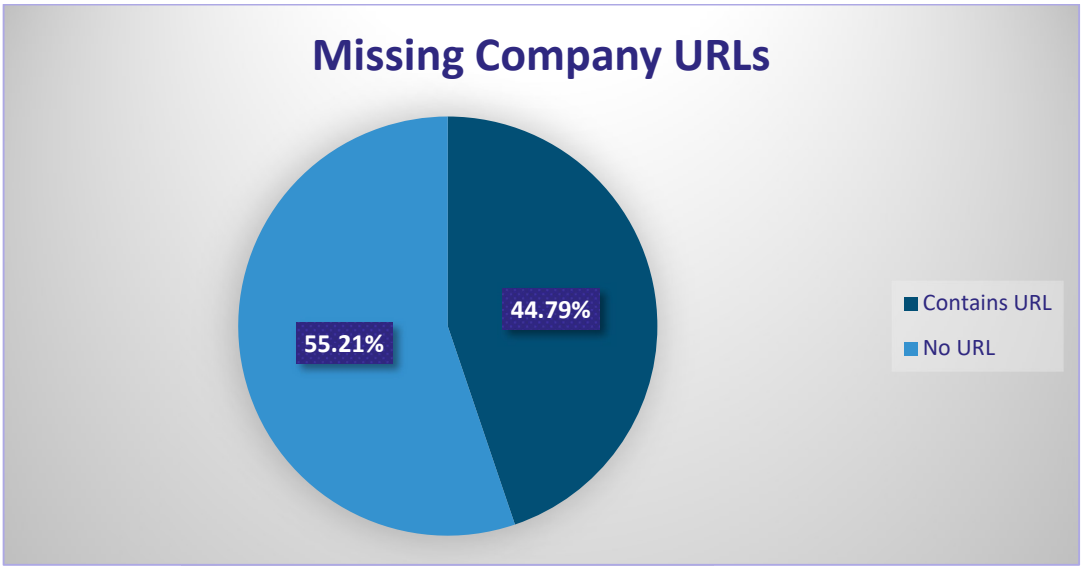
As mentioned earlier, we are using regex to check the pattern of our URL. The challenge we faced with that was that the regular expression we are using only covers the most commonly used types of URLs with domain names. If there are some more complex cases in the database, our regex might not catch it.

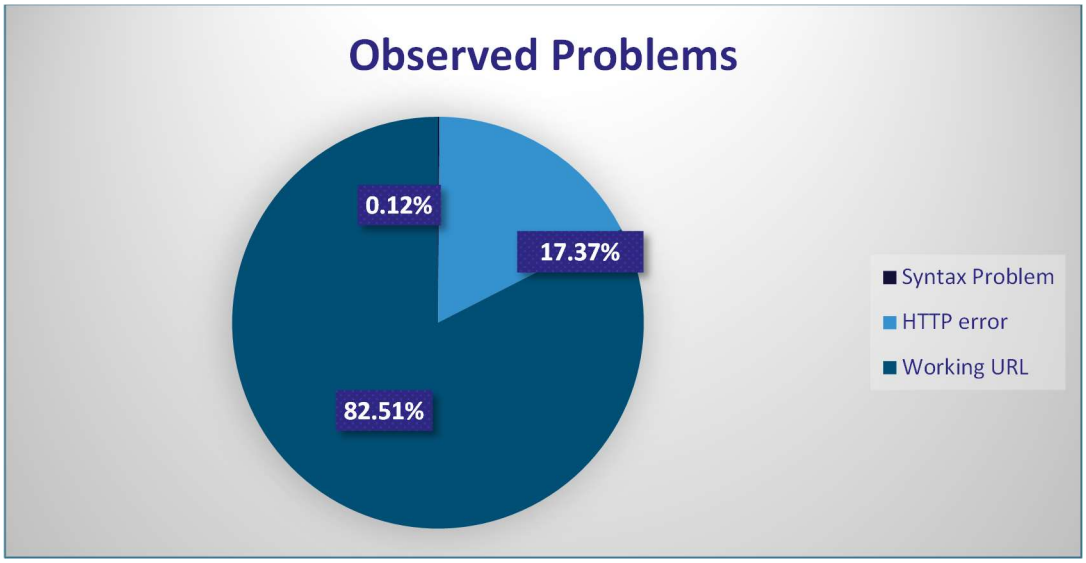
### 3.4.2. HTTP problems

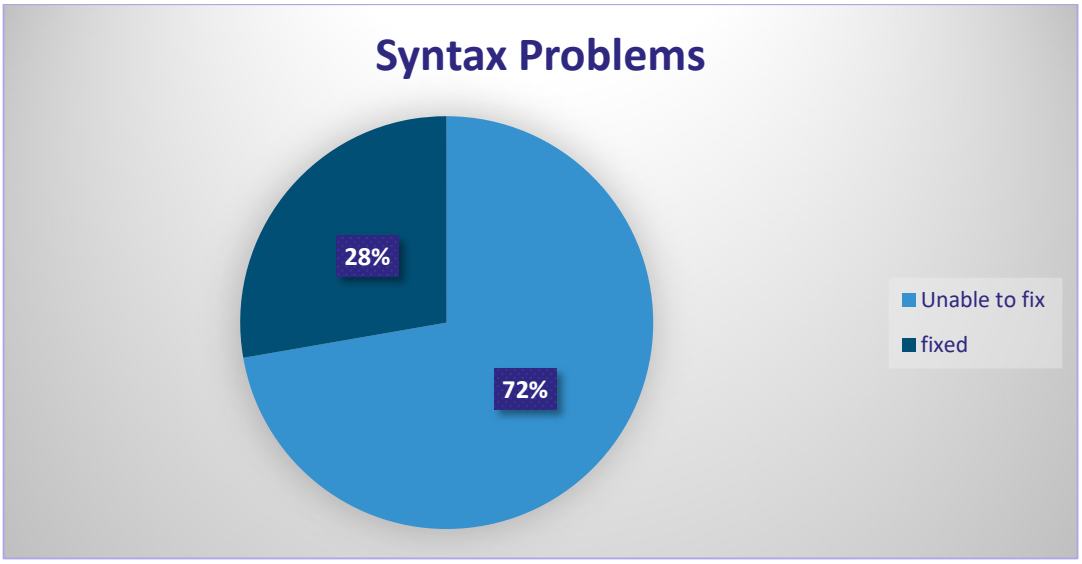
For analyzing URLs with http errors, we used a python package called requests. The requests library allows us to get data from a specified resource which would be the URL we are passing. We can also set a timeout, which is the maximum amount of time we will be waiting to get a response from the server. Some URLs will take up all the allowed time and the next request must wait for the first one to be done. And that does take a lot of time if most of the URLs take a long time to receive a response. For our case it took about 40 min to get the status code of a 1000 URLs and since we have

about 150,000 data in our database, it would be very time consuming. To solve our problem, we used a python package called thread pool executors. ThreadPoolExecutors provides multiple threads and uses these threads to perform tasks in a concurrent fashion. Adding threading to our implementation helped to drastically improve the speed of our program. By using multiple threads, we can speed up applications which face an input/output-based bottleneck. While one HTTP request is waiting for a response, another thread can be spawned to continue to make another request.

# 4. Statistical analysis of known problems in the database






# 5. Correcting data

## 5.1. Techniques used for correcting URLs with syntax problems

After analyzing data and finding URLs with syntax problems, we attempt to correct them.

## 5.2. re.sub

When attempting to fix the syntax problems, we used a python package called re. We have a series of conditions listed and based on those conditions we will replace any undesired patterns with the patterns that we want. We will do a second round of checking the syntax after correcting it and if it passes then we would add it to the csv.

## 5.3. Techniques used for correcting URLs with HTTP error

After analyzing data and finding URLs with HTTP errors, we consulted our client and our client mentioned that he only wanted us to take care of URLS with status code 404 or URLs that return -1 because our code returns -1 when the site can't be reached.
The approach we took for correcting URLs with those status codes was to remove them from the database and find a new working URL for the said business.

### 5.3.1. Appending

The first approach we took when looking for a working URL was to append the company name with a top-level domain, scheme and a third level domain (WWW.). Example: 'https://www.company name.com'. After appending the company name with the most common top-level domains, we will filter them and choose one by checking the status code of each. If their status code is not equal to 404 or -1, then they will be added to the list of potential URLs.

### 5.3.2. Using google search package to retrieve links from the web

After appending URLs, the next step, we took was to see one python package called Google-search to find a list of specified number of potential URLs. Using python package google search we can get results of google search from the python script. We can get links to first n search results. After retrieving those URLs, we add them to the list of potential URLs.

### 5.3.3. Scraping

Now that we have filled the list of potential URLs, the next step we followed was to do some web scraping to extract data from the webpage to see if these URLs have the correct contents. We count how much information from the BBB database matches the

information on the webpage. In the end, we chose the URL for the most match and append/update it to csv file.

## 5.4. Challenges with correcting URLs

### 5.4.1. Syntax problems

As mentioned above, we are fixing syntax problems by including various conditional statements. The issue we have with that is that there might be some syntax issues that we haven't considered including in our conditional statements.

### 5.4.2. Identifying the correct URL

We ran into some difficulties while trying to pick the correct URL for the company.

#### 5.4.2.1. Outdated information

Some information on the website and BBB's database wouldn't match because either information in the database or company website haven't been updated. If information is not matching, then our program will misclassify the correct URL as incorrect.

#### 5.4.2.2. Rating sites

When we are using the google search package, sometimes we get rating sites as URLs and these rating sites could be misclassified as the company URL because the information can match while scraping. The solution we had for this was to include the most well-known rating sites and avoid them. But there still might be some rating sites we haven't included in our list since there are many.

# 6. Things we would do differently if we had more time

## 6.1. Find a more reliable methods to assign weights to attributes

While web scrapping we are also keeping count of how much information in the database has matched with the information on the website. The information we picked was company name, address, city, state, email, and phone number. We assumed that the company name would be more important because it is distinct and is probably most likely to be mentioned on a website, so we gave it more weight. If we had more time, instead of assuming, it would have been more reliable if we used some type of statistical calculations like logistic regression to determine the weight for each attribute.

## 6.2. Include more cases for syntax errors

As mentioned above we have various conditional statements to fix syntax errors. If we had more time, we would have added more conditions to further decrease the number of syntax errors found in the database.