

**School of InfoComm Technology**

**Distributed Data Pipelines**

Diploma in Data Science (DS)

October 2023 Semester

**INDIVIDUAL ASSIGNMENT 2**

(40% of Distributed Data Pipelines Module)

**Deadline for Submission:**

**4th Feb 2024 (Sunday), 2359 Hours**

|  |  |  |
| --- | --- | --- |
| Student Name | : | Clayton Thia |
| Student Number | : | S10243162F |

**Penalty for late submission:**

10% of the marks will be deducted every day after the deadline.

**NO** submission will be accepted after 11th Feb 2024, 23:59.

Contents

[1. Problem Statement Formulation 3](#_Toc157822485)

[2. Preparation 4](#_Toc157822486)

[2.1 Google Drive Input 4](#_Toc157822487)

[2.2 Select 5](#_Toc157822488)

[2.3 Formula 5](#_Toc157822489)

[3. API Call 6](#_Toc157822490)

[3.1 Download 6](#_Toc157822491)

[3.2 Select 6](#_Toc157822492)

[3.3 JSON Parse 6](#_Toc157822493)

[3.4 Sample 7](#_Toc157822494)

[3.4 Text to Columns 8](#_Toc157822495)

[3.4 Cross Tab 8](#_Toc157822496)

[4. Data Display 9](#_Toc157822497)

[4.1 Multi-Field Formula 9](#_Toc157822498)

[4.2 Select 10](#_Toc157822499)

[4.3 Google Drive Output (enhancement) 10](#_Toc157822500)

[5. Enhancements 11](#_Toc157822501)

[5.1 Further Data Preparation 11](#_Toc157822502)

[5.2 Visualizations with Power BI 14](#_Toc157822503)

[6. Summary 16](#_Toc157822504)

[7. Reflection 17](#_Toc157822505)

[7.1 Possible improvements 17](#_Toc157822506)

[7.2 Skills learnt | could have learnt better 18](#_Toc157822507)

## Problem Statement Formulation

In the world of public transportation, knowing when the bus will arrive is crucial for commuters who take the bus as part of their daily lives. Dependable and timely data can make daily commutes smoother and less stressful.

The ability to provide accurate bus arrival times is extremely valuable to both individuals and companies. For organizations, this functionality enhances operational efficiency by enabling better resource allocation and scheduling. Employees may more efficiently arrange their commutes, which lowers absenteeism and tardiness and maximizes staff productivity. For individuals, accurate bus arrival times help to arrange their daily activities more effectively by reducing uncertainty and saving valuable time.

For this assignment, I have chosen the “Clementi Stn Exit B” bus stop. It is a bus stop of importance to Ngee Ann polytechnic as it is part of the commutes of not just regular citizens, but many of its students as well. Due to its significance, the bus stop is typically very crowded in the mornings to early afternoon and commuters are always fighting for a spot in buses. Knowing when buses arrive will allow students to position themselves more strategically such that they will be among the first few to secure a spot on the bus. As a result, students are more likely to get to school punctually, reducing lateness and potentially preventing them from falling victim to grade caps. Peak hours can be squeezy, even more so when groups of people are trying to get from one end to the other. Implementation of the model will lead to reduced overall movement in the bus stop.

Of course, members of the public also stand to benefit from reliable bus timings as they facilitate improved time management by enabling people to schedule their trips more effectively and spend less time waiting at bus stops. Due to its consistency, there is less ambiguity, which lessens the anxiety and annoyance that come with erratic public transportation timetables. For example, an individual who wants to maximize time spent with family or friends must catch the last bus home as he/she lives far away. An accurate bus arrival timing would allow the individual to safely get home for the night.

In conclusion, this model provides value by reducing lateness and providing safe and trustworthy timings that enable commuters to get where they want, when they want.

## Preparation

A diagram of a power supply system

Description automatically generated with medium confidence

*Fig 2.1 Screenshot of Preparation*

The first part of the Alteryx workflow is the preparation segment, where the model is set up to provide the appropriate Application Programming Interface (API) call based on the bus stop input. The API this workflow will be calling is LTA’s Bus Arrival Timings API in DataMall to load real-time stream data. This portion contains only 4 tools, excluding the Browse tool. The tools are, in sequential order, Text Input, Select, Formula, and Download.

### 2.1 Google Drive Input

Google Drive Input is a tool for reading data from a google drive. This workflow uses this tool to read data important for performing the API call – the API URL, its authentication key, and the target bus stop code. The target bus stop can be changed by simply altering the corresponding value in google sheets. Additionally, the tool has been configured to only include the first row of the sheet it is reading from. This is to prevent blank rows from entering the workflow and messing up the pipeline.

Alternatively, the Text Input tool could be used in place of Google Drive Input.

A screenshot of a computer

Description automatically generated

*Fig 2.2 Data in Google Sheets in Google Drive*

Text Input allows for the workflow user to manually input data which will then be stored within the workflow. It can be used to create columns in which the API URL, authentication key and bus stop code are entered.These columns will be used when signaling the API for its response.

### 2.2 Select

Performing an API call to LTA’s API for bus arrival timings requires two things – URL, and authentication key. The URL must be in a specific format where the target bus stop code is appended at the end.



*Fig 2.3 Sample API URL*

At this current point in the workflow, the URL value is yet to be in proper format as it is missing the bus stop code portion. Before I could append the bus stop code to the URL*,* I had to change the column’s data type to a string as Alteryx does not automatically convert numbers to string when concatenating strings. The Select tool was used as it allowed me to change the data type of the column containing the bus stop code.



*Fig 2.4 Error from attempting to concatenate a non-string value*

### 2.3 Formula

The Formula tool creates or updates columns by using expressions to perform a broad variety of operations. Using this, I appended the bus stop code into the URL - completing the setup for API calls. This tool was used instead of Multi-Field Formula as I was only executing expressions on one column.

## API Call

A diagram of a diagram

Description automatically generated

*Fig 3.1 Screenshot of API Call*

The next section of the workflow is the API call segment, where the output from the LTA’s API is downloaded into the model and undergoes the necessary operations to transform the data into an appropriate format, where each row represents a different bus service number, and each column represents the different features. This will be crucial for the next section where the data is displayed. Excluding Browse, 6 different tools. The tools are, in sequential order, Download, Select, JSON Parse, Sample, Text to Columns, and Cross Tab.

### 3.1 Download

The download tool retrieves data from a specified URL, making it the key component behind many API calls. This tool takes in important information like the API URL and authentication key before sending a signal to the API. The URL is necessary for the tool to know which API should be pinged, and the authentication key is needed for security clearance to access the API’s response. The result of this tool is the import of data from LTA’s API into the workflow where the data is stored in one single field, in one single row.



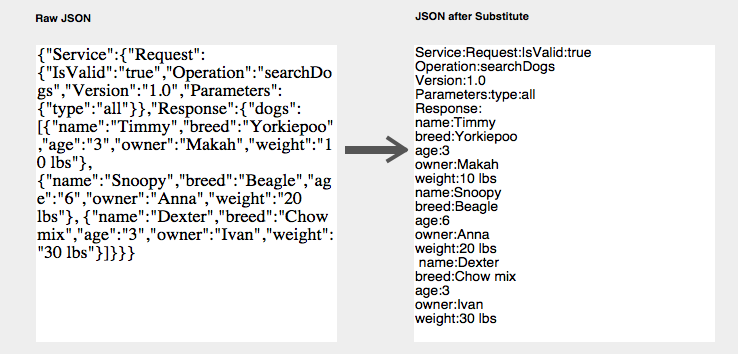
*Fig 3.2 API response*

### 3.2 Select

The Select tool allows for column management processes, such as renaming, adding, and removing, as well as changing the data type of columns. In the API Call section, the Select tool is used primarily to remove unnecessary columns. Removing such columns whenever possible makes the data flow more transparent and easier to understand for anyone reviewing the workflow (e.g. you, the lecturers), and reduces the computational load on subsequent tools by enabling the workflow to handle a more focused set of columns.

### 3.3 JSON Parse

API responses usually come in the form of a JavaScript Object Notation (JSON) object, and LTA’s API is no exception. The JSON object’s data is structured in a way that resembles lists and dictionaries in Python. The JSON Parse tool separates JSON text into a table schema, and in this workflow, it separates the different features in the JSON object into separate rows. This tool was used because it can efficiently handle nested structures by extracting and converting JSON data into tabular format.



*Fig 3.3 Sample result of JSON Parse*

*A screenshot of a computer

Description automatically generated*

*Fig 3.4 Separation of data into rows*

### 3.4 Sample

This tool limits the data stream to a number, percentage, or random set of records. It is useful for removing or looking at a subset of data within the data stream. In this workflow, I use the Sample tool to skip the first 2 rows of the data, which are values for the metadata and bus stop code as seen in figure 3.4. Removing these rows are important for later tools as irrelevant data may interfere with processing.

The Select tool was not applicable here as it only deals with columns, and does not have functionalities regarding management of data records.

### 3.4 Text to Columns

This tool splits the text from one field into separate rows or columns, using a specified value as the delimiter on which it splits the text. In the workflow, the tool splits the text under *JSON\_Name* into 3 different sections as shown in figure 3.5. The result of the tool is the creation of new columns *1*, *2*, and *3*.

A screenshot of a computer

Description automatically generated

*Fig 3.5 Result of Text to Columns*

Splitting is essential as this allows me to get the necessary fields to transform the data into proper table format for displaying – namely column *2* which separated each feature by the different bus service numbers. After splitting, the *JSON\_Name* and *1* columns were removed using the Select tool.

### 3.4 Cross Tab

The Cross Tab tool pivots the orientation of data in a table by moving vertical data onto a horizontal axis and summarizing the selected data. In other words, this tool could convert the rows in the data which currently represent the different features of each bus service number into columns – the ideal table structure. The data is grouped by column *2* which represents the bus service number. A Select tool is then used to remove and arrange columns.

A screenshot of a computer

Description automatically generated

*Fig 3.6 Result of Cross Tab*

## Data Display

A screenshot of a computer

Description automatically generated

*Fig 4.1 Screenshot of Data Display*

The final part of the workflow is the Data Display section, where the data is tidied up and exported for displaying purposes. This section contains 3 unique tools excluding Browse – in sequential order, Multi-Field Formula, Select, and Google Drive Output.

In between the API call and Data Display sections, further enhancements were made that will be addressed in the next part of the report. New or unknown columns can be attributed to said enhancements.

### 4.1 Multi-Field Formula

The Multi-Field Formula tool has the same functionalities as the Formula tool. However, it is more suited towards executing the same expression on multiple columns at once. This tool was used to replace values in *\_Load* and *\_Type* columns, which represent the capacity and type of bus respectively, to much more interpretable forms. For example, the ‘SD’ value in *\_Type* columns to ‘Single’ to represent single decker buses.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

*Fig 4.2 Before (left) and after (right) Multi-Field Formula*

### 4.2 Select

The Select tool was used to remove unnecessary columns and rename columns such that they were easier to interpret. For example, renaming *NextBus\_Load* to *1st Bus Capacity*.

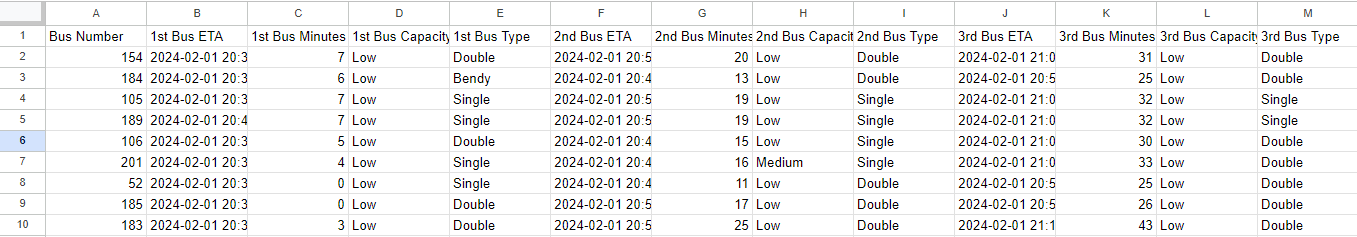
A screenshot of a computer

Description automatically generated

*Fig 4.3 Renaming columns*

### 4.3 Google Drive Output (enhancement)

For the final display, the Google Drive Output tool was used. Google Drive Output has the exact opposite functionality of its counterpart and the very first tool in the workflow - Google Drive Input., where it writes data from the workflow into a file from Google Drive. Data from the workflow’s output was entered into a Google Sheets file using this tool. Google Sheets was used as it can be connected to Power BI, which will be used in a later enhancement.



*Fig 4.4 Output into Google Sheets*

This data structure is now displayed in perfect structure and format for use in Power BI, and can be viewed and played around with this [link.](https://docs.google.com/spreadsheets/d/1eDORRJiRssZbmHjfd4ohr4spKM-jJUBZ3Md8nZPcu3c/edit?usp=sharing)

## Enhancements

The enhancement for this module would be the use of Power BI to display the bus arrival times and other data with visualizations. Before displaying the data as visuals, however, further data preparation is required.

Between the API Call and Data Display section, I performed said further data preparation where I aimed to create new columns that reflect the minutes between the current time and the estimated arrival time of buses. To do so, the *\_EstimatedArrival­* columns needed to be converted to DateTime and proper formulae had to be executed. The tools used in this section, in sequence, are Multi-Field Formula, DateTime, and then Multi-Field Formula again.

### 5.1 Further Data Preparation

A green and white logo

Description automatically generated with medium confidence

*Fig 5.1 Converting to DateTime*

#### 5.1.1 Multi-Field Formula

The data type of the column with DateTime data was not DateTime. This was because the values, as seen in figure 5.2, were in a format that Alteryx could recognize on default. Additionally, there was a time-zone offset portion of the text that when converted to DateTime, added the time-zone offset to the actual time – changing it such that it did not reflect Singapore’s local time. This tool allowed me to remove the time-zone part of the text such that it can be properly converted to DateTime further down the processing pipeline.

A screenshot of a computer

Description automatically generated

*Fig 5.2 Values in supposed DateTime column*

#### 5.1.2 DateTime

This tool allows for the conversion of column data types to and from DateTime, given the proper format. Since the text was not in a default format Alteryx could not recognize, a custom format had to be specified.

A screenshot of a computer

Description automatically generated

*Fig 5.3 Custom format for DateTime text*

|  |  |
| --- | --- |
| **%Y** | Represents the four-digit year |
| **%m** | Represents the two-digit month |
| **%d** | Represents the two-digit day |
| **T** | Indicates the start of the time portion |
| **%H** | Represents the two-digit hour in 24-hour format |
| **%M** | Represents the two-digit minute |
| **%S** | Represents the two-digit second |

*Fig 5.4 Custom format values*

Using the above format in figure 5.3, the columns were successfully converted to DateTime. This means that DateTime expressions such as DATEDIFF() can now be accessed and used on the columns.

#### 5.1.3 Multi-Field Formula

A Multi-Field Formula tool was used once again to complete the objective for further data preparation – the creation of a column that showed the remaining minutes before the estimated arrival time of the buses. Using a DateTime expression, DATETIMEDIFF(), the column was created, and its values were calculated. The expression also accounts for if there are no next buses. For example, if the value in ­*NextBus3\_EstimatedArrival* is null, the value in the new column will be null as well. This is done as the tool will output a 0 if the DateTime column is null. Additionally, the expression uses a MAX() function as the result of DATETIMEDIFF() could be -0 sometimes.

A screenshot of a computer program

Description automatically generated

*Fig 5.5 Expression for calculating of values for new column*

Ensuring null values were properly placed was important as it may interfere with visualization in Power BI.

A screenshot of a bus schedule

Description automatically generated

*Fig 5.6 Result of formula*

With this, the further data preparation portion is completed.

### 5.2 Visualizations with Power BI

The Power BI file retrieves data from the Google Sheets file that the workflow had output to. Using the retrieved data, a dashboard was created.

A screenshot of a computer

Description automatically generated

*5.7 Power BI Dashboard*

The dashboard features a keypad-like filter on the left for the user to toggle between the different bus services, depending on which bus timings they’d like to view. On the right there are 3 rows displaying the minutes to arrival, capacity, and type of the bus. From top to bottom, the rows correspond to the first, second, and third buses respectively. Using this dashboard, the user can quickly see the most important details of any bus service number.

The dashboard can also be used for other bus stops. For example, the bus stop at block 946A at Hougang. Visuals on the dashboard will change accordingly to the bus stop code as seen in figure 5.8.

A screenshot of a computer screen

Description automatically generated

*Fig 5.8 Dashboard for a different bus stop code*

Dashboards also include mobile support which does not cut out on information, albeit the UI is less clean.

A screenshot of a phone

Description automatically generated

*Fig 5.9 Mobile layout for Dashboard*

The dashboard can be accessed through [this link](https://app.powerbi.com/links/fmU2B3Tigj?ctid=cba9e115-3016-4462-a1ab-a565cba0cdf1&pbi_source=linkShare&bookmarkGuid=5bbaa444-04f9-42fc-8f32-be8dba22a192), and the Power BI file will be included in submission.

## Summary

The model value-adds to users by relaying important information to them like the bus arrival timings, as well as their capacity. This allows for users to schedule their trips effectively to get where they want, when they want.

Throughout the creation of the model, important findings were made, and not all of them were documented in any of the report sections above. Below are my findings.

**Alteryx**

1. **Alteryx automatically initializes appropriate data types but does not convert when performing expressions.** An example of this would be in the preparation section where an error relating to data types was encountered when appending the bus stop code to the end of the URL. The error code was due to the bus stop code not being a string data type and such it was not possible for it to be appended to the end of another string. This meant that before performing string expressions that involve columns with numbers, data types had to be checked beforehand to ensure no errors arose.
2. **Browse tools slow down the workflow.** The Browse tool is useful for the viewing of data profiles at specific points in the workflow, allowing for effective reviewing of tool results. However, Browse creates a temporary .yxdb file which holds data and therefore can slow down the workflow when used too often. As a result, most of the Browse tools were removed from the workflow after finalizing it. The lesson learned here is to use the tool in moderation.
3. **Tool configurations may reset.** When a tool receives a blank data flow – meaning it doesn’t receive any data – its configurations may reset. An error in one tool may lead to a blank data flow downstream and may cause a cascade of configuration resets. This is an extremely annoying and troublesome phenomenon that can only be prevented by careful double-checking of the relationships between the data, workflow tools, and expressions.
4. **Alteryx supports additional packages.** Examples of such packages would be the Power BI and Google Drive connector which supports integration with the relevant platforms. This can bring about benefits like data source variety and enhanced reporting and visualizations.

**Data**

1. **The API response can and will get smaller in size.** As the last buses finish their shift, there aren’t any next buses for the API to return information about. This leads to null values in related columns, and the vanishing of an entire chunk of data about a service number when the last bus is done with its path. This meant that the workflow cannot be properly reviewed and worked on approaching midnight and wee hours unless data input is hardcoded into the workflow.
2. **Columns containing DateTime information are not in proper DateTime format.** The columns refer to the *\_EstimatedArrival* columns where a custom format had to be specified for the DateTime tool to work, and each *\_EstimatedArrival* column needed its own DateTime tool to convert its data type into DateTime. This means that if the API is enhanced to provide data for more incoming buses, more DateTime tools will be needed for conversion to DateTime, leading to higher computation times.

## Reflection

### 7.1 Possible improvements

**Real-time updates**

The first improvement to be made would be the **implementation of real-time updates.** Currently, the Power BI dashboard occasionally refreshes and retrieves data from Google Sheets, which is only updated when the workflow runs. For real-time updates to be possible, the workflow must be scheduled to run periodically, meaning that an Alteryx Server license is required. Unfortunately, organizational access restricts my Alteryx account from getting an Alteryx Server license. To make matters worse, I could not apply for a free trial with my personal account, as I did not receive any registration / verification emails after numerous attempts of applying for the free trial in Alteryx Cloud.

Finding a workaround to get an Alteryx Server will boost how much the solution value-adds to the public by providing automatic real-time updates.

**Automation**

As stated before, a possible improvement to the solution would be to schedule the workflow. This could be done through automation with python, where we use python modules like PyAutoGUI and PyGetWindow to get user input on which bus stop code the user would like to target, and control computer input to run the workflow. However, this essentially means that the computer is “hijacked” by the program for as long as it takes to run the workflow. Moreover, the workflow must be run every minute or so to provide live data.

Therefore, I did not implement this into my solution as I believed that automation is something that should value-add to whatever it is being added to, and that would not be possible if the host’s usage of their machine was interrupted every minute.

**Connecting directly to Power BI**

The third improvement would be to **connect the workflow directly to Power BI**. The pipeline of data is from the workflow to Google Sheets and finally to Power BI. Linking the workflow to Power BI can cut out the middleman, Google Sheets, and potentially speed up computational time. The Power BI connector tool was used, but organizational data security settings interfered with the connecting process, rendering that approach null. Hence, Alteryx could not connect directly to Power BI.

Successful connection would improve the model by enabling it to update data sooner.

### 7.2 Skills learnt | could have learnt better

Describing the data pipelining process

My comprehension of how data travels through different phases—from extraction to transformation to loading—has been reinforced by this module. I have also learnt to consider additional factors, such as data validation, scheduling and automation within the pipeline, as well as scalability and optimization. This strengthens my capacity to contribute to effective data management by giving me the knowledge necessary to optimize and streamline the flow of data within an organization.

List the basic components of Hadoop, and understand their usages

My understanding of distributed processing and storage has expanded because of the studying of Hadoop's fundamental components. In the age of big data, knowing how to use components like HDFS and MapReduce is important.

List the basic components of Apache Spark, and understand their usages

Learning the fundamentals of Apache Spark has been especially insightful. Acquiring knowledge about RDDs and using Spark in my assignment has enhanced my machine learning, data processing, and analysis toolkit. A skill I could have learned better would be the management of computing resources, as in the first assignment, my notebook took around half an hour to run despite not dealing with huge datasets or performing many transformations.

List the basic components of Alteryx, and understand their usages

The best part of this module has been learning about Alteryx and its basic elements. By developing an Alteryx workflow to show bus arrival times, the platform's robust data preparation features and intuitive interface were demonstrated. The ability to effortlessly integrate different data manipulation tools and perform drag-and-drop operations has enabled me to efficiently design and implement workflows. To improve even more, though, I think that practicing in more complicated situations will increase my proficiency with Alteryx by forcing me to explore the variety of tools within the Alteryx workflow designer. This would lead to flexibility with the software in various situations and prepare me for real-world situations.

Design and deploy a data pipelining infrastructure based on a company’s business requirements

When I consider the learning objectives, I see that the experience has given me useful abilities for developing and implementing data pipelining infrastructures. Developing an Alteryx workflow for the real-world use of showing bus arrival times has been an essential part of consolidating the theoretical knowledge learned in this module. Continuous practice and exposure to a variety of data scenarios will definitely boost my confidence and competence in applying these skills to actual business requirements in the real-world.