Solution Sheet: Parsing and Transforming Address Data Using Alteryx

Overview

This project demonstrates how to parse and transform unformatted address data using Alteryx Designer. By utilizing tools such as **Find & Replace**, **Text to Columns**, and **RegEx Parse**, the workflow separates address components into distinct fields for **City**, **State**, and **ZIP code**. The structured data is then organized into a clean and usable format for further analysis.

Step-by-Step Solution

Task 1: Import Address Data Using Input Tool

- Open Alteryx Designer and create a new workflow.
- Drag and drop the **Input Data** tool onto the canvas.
- Configure the tool to import the address dataset by selecting the appropriate file.
- Run the workflow to load the data into Alteryx for processing.

Task 2: Extend String Size Using Select Tool

- Add a **Select** tool to the workflow.
- Connect it to the output of the **Input Data** tool.
- Extend the string size for the address field to ensure it can handle longer entries.

Task 3: Select Required Fields Using Select Tool

- Use the **Select** tool to keep only the necessary fields, such as **First Name**, **Last Name**, and **Address**, for the project.

Task 4: Import Additional Data for Road Type Suffix

- Drag and drop another **Input Data** tool to import the road type suffix dataset.
- Configure the tool to load this dataset, which will be used to add a pipe ("|") as a delimiter for road types.

Task 5: Add Pipe Delimiter Using Find & Replace Tool

- Add a **Find & Replace** tool to the workflow.
- Configure the tool to match road types in the **Address** field and add a pipe ("|") as a delimiter after each road type.

Task 6: Split Address Data Using Text to Columns Tool

- Add a **Text to Columns** tool to the workflow.
- Split the **Address** field into two columns using the pipe ("|") as a delimiter.

Task 7: Parse City and State Using RegEx Tool

- Add a **RegEx** tool to parse the **City** and **State** fields from the address data.
- Use the following RegEx formula:

scss

```
^\s(.+)\s(\u\u).*
```

- This formula extracts the city name and two-letter state code from the address.

Task 8: Parse ZIP Code Using RegEx Tool

- Add another **RegEx** tool to parse the **ZIP Code** field.
- Use the following RegEx formula:

ruby

```
^.+(\d\d\d\d\d)$
```

- This formula captures the 5-digit ZIP code from the address data.

Task 9: Remove Unnecessary Fields Using Select Tool

- Use another **Select** tool to retain only the fields required for the final output: **First Name**, **Last Name**, **City**, **State**, and **ZIP Code**.

Task 10: Viewing Results Using Browse Tool

- Add a **Browse** tool to the workflow.
- Connect it to the output of the **Select** tool.
- Run the workflow to view the transformed data with separated columns for **City**, **State**, and **ZIP Code**.

Task 11: Save the Workflow for Future Use

- Save the completed workflow by using the **Save As** option in Alteryx Designer.
- Assign a meaningful name and location to the workflow for easy access and reuse.

Conclusion

This workflow transforms unformatted address data into a structured table with separate fields for **City**, **State**, and **ZIP Code**. The project demonstrates effective use of Alteryx tools, including **Find & Replace**, **Text to Columns**, and **RegEx Parse**, to address data parsing challenges. The resulting workflow is efficient, reusable, and provides well-organized data for downstream applications.