# TAT Dashboard Data Processing Pipeline Documentation

**Overview**

This pipeline is designed to automate the ingestion, cleaning, transformation, and uploading of data from various providers into a Sandpit Table. Providers submit data each month for the previous three months in two formats: **flex** (subject to revisions) and **freeze** (finalized data). The pipeline processes this data, replaces older flex data with freeze data, and handles the appropriate logic for new flex data in Sandpit.

**1. Pipeline Workflow**

1. **Monthly Data Submission**:
   * Each month, providers submit data for the last three months. One month's data is freeze data, while the other two are flex data.
   * Example for October 2024:
     + **Freeze**: July 2024 data.
     + **Flex**: August 2024 and September 2024 data.
2. **Ingestion and Processing**:
   * Files are read from their respective provider folders.
   * The file names contain metadata (submission date, provider code, month-year, and data type) that is extracted to add columns like submission\_date, provider\_code, month, year, and data\_type to the data.
3. **Data Upload**:
   * Freeze data replaces old flex data for the same month.
   * Flex data is checked if it already exists in the SQL Server table:
     + If it exists, it's replaced.
     + If it doesn't exist, it's appended.
4. **SQL Server Upload**:
   * Data is uploaded or updated in a Sandpit Table following the logic mentioned.

**2. File Naming Convention**

The pipeline relies on a standardized file naming convention to extract metadata. The expected format is:

[submission\_date]\_[provider\_code]\_[month-year]\_[data\_type].csv

**Example:**

* 20240924\_DID\_RAL\_Jun24\_Freeze.csv
  + Submission Date: 2024-09-24
  + Provider Code: RAL
  + Month-Year: June 2024
  + Data Type: Freeze

**3. R Script Components**

**3.1 Initialization and Setup**

**Dependencies**:

* tidyverse: For data manipulation.
* lubridate: For date handling.
* janitor: For column name normalization.
* DBI and odbc: For connecting and interacting with SQL Server.

**Install Necessary Packages**:

install.packages(c("tidyverse", "lubridate", "janitor", "DBI", "odbc"))

**Source the functions:**

The functions used in the pipeline are stored in a separate R file called “functions” in the src folder.

**3.2 Data Ingestion**

Files are ingested using list.files() to locate all CSV files in provider folders. The metadata (e.g., provider\_code, month, year, data\_type) is extracted from the file names using regular expressions.

**3.3 Data Cleaning and Transformation**

The column names from the CSVs are normalized to lowercase and spaces are replaced by underscores using the janitor::clean\_names() function.

New columns are added for month, year, and other metadata extracted from the file names.

**3.4 SQL Upload Logic**

The SQL logic is divided into two steps:

* **For Freeze Data**:
  + Delete old flex data for the same provider, month, and year from the SQL table.
  + Insert new freeze data.
* **For Flex Data**:
  + Check if flex data for the same provider, month, and year exists.
  + If it exists, delete and replace it.
  + If it doesn't exist, append the new data.

**4. Error Handling**

Error handling is implemented using tryCatch() to ensure that failures during data processing, file reading, or SQL execution are logged and handled without stopping the entire process.

**Example of Error Handling in SQL Execution:**

r

Copy code

tryCatch({

dbExecute(con, delete\_query)

}, error = function(e) {

warning("Error deleting data for ", provider\_code, ": ", e$message)

})

In case of an error, the process will continue to handle the next file or record, and warnings will be logged.

**5. Database Requirements**

The Sandpit table where the data is uploaded should have the following structure:

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| provider\_code | VARCHAR(3) | Provider identifier code |
| month | VARCHAR(3) | Short month name (e.g., 'Jan') |
| year | VARCHAR(4) | Year (e.g., '2024') |
| data\_type | VARCHAR(5) | 'Freeze' or 'Flex' |
| submission\_date | DATE | Date when data was submitted |
| (other columns) | (depends) | The actual data fields from the CSVs |

**6. How to Use**

1. **Set Up Project**:
   * Make sure you have the required R packages installed. **Use renv::init() to ensure that the correct versions of the packages are installed.**
   * Set up the file directories where provider data will be stored.
2. **Run the Script**:
   * Ensure that the input\_dir variable points to the folder containing provider submissions.
   * Execute the R script in an R environment or as a Quarto document.
3. **Uploading Data**:
   * The script will connect to your Sandpit Table, process the data, and update the table with freeze and flex data.

**7. Troubleshooting**

**Connection Issues:**

* If you encounter connection issues with SQL Server, check your credentials.
* Ensure the odbc driver is correctly installed and configured.

**File Processing Errors:**

* If a file does not follow the naming convention, the script might fail to extract metadata. Ensure that all files are named according to the format.

**SQL Insertions Failing:**

* If insertion fails, check the Sandpit table schema to ensure it matches the incoming data structure.

**8. Maintenance**

**Data Directory Structure:**

* Ensure that the folder structure where provider data is stored remains consistent each month.

**Extending the Pipeline:**

* If additional data transformations are required, these can be added in the process\_provider\_files() function.

This documentation provides a complete guide on how to set up, run, and maintain the data processing pipeline, ensuring smooth operation month-to-month.