SYED MUHAMMAD IRTIZA ZAIDI - 23973

[Company Name]

ETL PROCESS

Table of Contents

[Introduction 2](#_Toc137637564)

[Code Explanation 2](#_Toc137637565)

[Explanation of ETL Process: 2](#_Toc137637566)

[The provided code implements the ETL process as follows: 3](#_Toc137637567)

[DataFrame Details and ETL Implementation in the Code 4](#_Toc137637568)

[Python Libraries Used: 6](#_Toc137637569)

[Conclusion 6](#_Toc137637570)

# 

# Introduction

The provided code showcases an implementation of the ETL (Extract, Transform, Load) process using Python and the pandas library. ETL is a fundamental process in data integration and data warehousing, involving the extraction of data from various sources, transformation to make it suitable for analysis, and loading into a target destination. In this case, the code reads data from multiple text files in a directory, performs transformations on the extracted data, and presents the transformed data for further analysis.

# Code Explanation

The code begins by specifying the directory path containing the text files to be processed. It then utilizes the **read\_dir** function to extract the data from each text file within the specified directory. The extracted data is organized into a dictionary.

Next, the extracted data is converted into a pandas DataFrame (**df**) using **pd.DataFrame(dict)**. The DataFrame serves as the starting point for the subsequent ETL transformations. Several transformations are applied to this DataFrame, including replacing certain values, adding new columns, and removing duplicate rows.

The transformed data is further processed to create additional DataFrames, each focusing on specific aspects of the data. These DataFrames include **df1\_rejected\_rows**, **df1\_committed\_rows**, **df1\_row\_read**, and **result**. They represent subsets of the original DataFrame (**df**) based on different filtering criteria or grouping operations.

Finally, the code prints the contents of each DataFrame, displaying the results of the ETL process.

# Explanation of ETL Process:

The ETL (Extract, Transform, Load) process is a common approach used in data integration and data warehousing. It involves the following steps:

1. Extract: In this step, data is extracted from various sources, such as files, databases, APIs, or web scraping. The extracted data is typically in its raw form, preserving the original structure and format.
2. Transform: The extracted data is transformed to make it suitable for analysis and storage. This step includes tasks such as cleaning the data, filtering out irrelevant information, converting data types, aggregating data, and performing calculations. The transformations are applied to ensure consistency, correctness, and usability of the data.
3. Load: The transformed data is loaded into a target destination, such as a database, data warehouse, or data file. The data is organized and stored in a structured manner, facilitating efficient querying, reporting, and analysis.

## 

### The provided code implements the ETL process as follows:

1. Extraction:

* The code reads text files from a specified directory (dir\_path) using the read\_dir function.
* The content of each file is extracted using regular expressions in the read\_file function, retrieving specific information such as the number of rows read, inserted, updated, rejected, skipped, and committed.
* The extracted data is stored in a dictionary.

1. Transformation:

* The extracted data is converted into a DataFrame (df) using pd.DataFrame(dict),where each row of the DataFrame represents data from a single text file.
* Various transformations are applied to the df DataFrame, including replacing "0" with "zero", adding a new merged column, and removing duplicate rows.
* Additional DataFrames (df1\_rejected\_rows, df1\_committed\_rows, df1\_row\_read, and result) are created by filtering, grouping, and aggregating the data to extract specific insights.

1. Load:

* The transformed data is stored in different DataFrames (df1\_rejected\_rows, df1\_committed\_rows, df1\_row\_read, and result) for further analysis and visualization.
* The contents of each DataFrame are printed to display the results of the ETL process.

Overall, the code successfully demonstrates the ETL process by extracting data from text files, applying transformations, and storing the transformed data in DataFrames for further analysis.

## 

## DataFrame Details and ETL Implementation in the Code

1. **df** DataFrame:
   * Purpose: This DataFrame is created to store the raw data extracted from the text files.
   * Creation: The data is extracted using the **read\_dir** function, and then it is converted into a DataFrame using **pd.DataFrame(dict)**.
   * Transformations: An extra line (**df = df.drop(0)**) is present in the code, which drops the first row of the DataFrame.
   * Output: The contents of the **df** DataFrame are printed using **print(df)**.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Process | Branch code | Rows read | Rows skipped | Rows inserted | Rows updated | Rows rejected | Rows comitted |
| 1 | CAE | BRN-5 | 230 | 0 | 200 | 0 | 30 | 230 |
| 2 | AHS | BRN-4 | 76 | 0 | 70 | 0 | 6 | 70 |
| 3 | ABS | BRN-4 | 139 | 0 | 130 | 0 | 9 | 130 |
| 4 | ACT | BRN-4 | 92 | 0 | 90 | 0 | 2 | 90 |
| 5 | ABS | BRN-1 | 68 | 0 | 67 | 0 | 1 | 67 |
| 6 | EEE | BRN-7 | 88 | 0 | 78 | 0 | 10 | 78 |
| 7 | AAA | BRN-4 | 30 | 0 | 29 | 0 | 1 | 29 |
| 8 | SAT | BRN-5 | 10 | 0 | 5 | 0 | 5 | 5 |
| 9 | ABS | BRN-7 | 100 | 0 | 92 | 0 | 8 | 92 |
| 10 | AHS | BRN-1 | 25 | 0 | 25 | 0 | 0 | 25 |
| 11 | MEO | BRN-1 | 11 | 0 | 11 | 0 | 0 | 11 |
| 12 | ACT | BRN-5 | 50 | 0 | 50 | 0 | 0 | 50 |

1. **df1** DataFrame:
   * Purpose: This DataFrame represents the transformed data after applying certain modifications to the **df** DataFrame.
   * Transformations:
     + Replacing "0" with "zero": **df1 = df.copy().replace("0", "zero")**. This transformation replaces all occurrences of "0" in the DataFrame with the string "zero".
     + Adding a new column: **df1 = df1.assign(MergedColumn=df1["Process"].astype(str) + "-" + df1["Branch code"])**. This transformation creates a new column named "MergedColumn" by combining the values of the "Process" and "Branch code" columns.
     + Removing duplicate rows: **df1.drop\_duplicates(inplace=True)**. This transformation drops any duplicate rows from the DataFrame.
   * Output: The contents of the **df1** DataFrame are printed using **print(df1)**.
2. **df1\_rejected\_rows** DataFrame:
   * Purpose: This DataFrame represents a subset of the **df1** DataFrame, containing only the rows where the "Rows rejected" column is not equal to "zero". It filters out the rows that were rejected during the data processing.
   * Transformations:
     + Removing rows where "Rows rejected" is "zero": **df1\_rejected\_rows = df1\_rejected\_rows.drop(df1\_rejected\_rows[df1\_rejected\_rows["Rows rejected"] == "zero"].index)**. This transformation drops rows where the value in the "Rows rejected" column is "zero".
   * Output: The contents of the **df1\_rejected\_rows** DataFrame are printed using **print(df1\_rejected\_rows)**.
3. **df1\_committed\_rows** DataFrame:
   * Purpose: This DataFrame represents a subset of the **df1** DataFrame, containing only the rows where the "Rows committed" column is not equal to "zero". It filters out the rows that were not committed during the data processing.
   * Transformations:
     + Removing rows where "Rows committed" is "zero": **df1\_comitted\_rows = df1\_comitted\_rows.drop(df1\_comitted\_rows[df1\_comitted\_rows["Rows committed"] == "zero"].index)**. This transformation drops rows where the value in the "Rows committed" column is "zero".
   * Output: The contents of the **df1\_comitted\_rows** DataFrame are printed using **print(df1\_comitted\_rows)**.
4. **df1\_row\_read** DataFrame:
   * Purpose: This DataFrame represents a subset of the **df1** DataFrame, containing only the rows where the "Rows read" column has a value greater than or equal to 100. It filters out rows with a low number of rows read.
   * Transformations:
     + Filtering rows based on "Rows read" condition: **df1\_row\_read = df1\_row\_read[(df1\_row\_read["Rows read"] >= 100)]**. This transformation keeps only the rows where the value in the "Rows read" column is greater than or equal to 100.
   * Output: The contents of the **df1\_row\_read** DataFrame are printed using **print(df1\_row\_read)**.
5. **Result DataFrame** (not explicitly named):
   * Purpose: This DataFrame represents the grouped summary of total rows read, inserted, and rejected based on the branch code.
   * Transformations:
     + Grouping and summing the rows read, inserted, and rejected: **result = df.groupby("Branch code")[["Rows read", "Rows inserted", "Rows rejected"]].sum()**. This transformation groups the data by the "Branch code" column and calculates the sum of "Rows read", "Rows inserted", and "Rows rejected" for each group.
     + Renaming columns: **result = result.rename(columns={"Rows read": "Total Rows Read", "Rows inserted": "Total Rows Inserted", "Rows rejected": "Total Rows Rejected"})**. This transformation renames the columns to provide more descriptive names.
   * Output: The contents of the **result** DataFrame are printed using **print(result)**.

## 

|  |  |  |  |
| --- | --- | --- | --- |
|  | Total rows read | Total rows inserted | Total rows rejected |
| Branch code |  |  |  |
| BRN-1 | 104 | 103 | 1 |
| BRN-4 | 337 | 319 | 18 |
| BRN-5 | 290 | 255 | 35 |
| BRN-7 | 188 | 170 | 18 |

## Python Libraries Used:

The code utilizes several Python libraries to perform various tasks. Let's discuss each library and its purpose:

1. **pandas**: The pandas library is a powerful tool for data manipulation and analysis. It provides data structures such as DataFrames and Series, along with a wide range of functions for data cleaning, transformation, and analysis. In the code, pandas is used extensively for reading, cleaning, and manipulating the extracted data.
2. **os**: The os module provides a way to interact with the operating system. It is used in the code to retrieve the list of files in a directory and join file paths.
3. **re**: The re module provides support for regular expressions in Python. It is used to extract specific patterns from the content of text files in the **read\_file** function.
4. **matplotlib.pyplot**: The pyplot module from the matplotlib library is used for creating various types of plots and visualizations. In the code, it is utilized to generate visual representations of the analyzed data.

## Conclusion

The presented code effectively demonstrates the ETL process by reading data from multiple text files in a specified directory, performing transformations on the extracted data, and producing transformed DataFrames. The transformations encompass replacing values, creating new columns, removing duplicates, filtering rows based on conditions, and grouping data. The resulting DataFrames provide valuable insights and facilitate further analysis of the processed data.

The ETL process implemented in this code serves as a crucial step in data integration and data preparation. By extracting, transforming, and loading data, organizations can convert raw and heterogeneous data into a standardized, consistent, and analyzable format. This process lays the foundation for data-driven decision-making, advanced analytics, and business intelligence.