**Project Summary: Fitbit ETL Pipeline**

***Overview:***

The Fitbit ETL (Extract, Transform, Load) Pipeline is designed to process data from Fitbit devices, perform various operations including extraction, transformation, and load into a destination data warehouse. The pipeline ensures efficient handling of Fitbit data for further analysis and insights.

***Components:***

Extract Phase:

1. *Library Import:*

Imports required libraries for data processing tasks, including data manipulation, visualization and database transactions.

1. *Inputer Function (inputer\_func):*

Prompts the user to input necessary information for data extraction and transformation. The dataset folder path and database credentials for both source and destination databases (Username, Password, Server name, Database) are the input details.

1. *Source Data Extraction Function (source\_data\_func):*

Extracts data from CSV files located in the specified folder path and returns a list of Pandas DataFrames, each containing data from a CSV file.

1. *Extraction and Staging Function (extract\_and\_stage\_func):*

Extracts the raw data from the list of data frames to a temporary/staging platform (e.g. Postgres Database). This function establishes a connection to the staging database using provided credentials, creates necessary tables in the database if they don't exist before loading extracted data into the corresponding tables. At this point, the data is staged for transformation.

Transform Phase:

Transformation Function (transformation\_func): This function contains other functions that performs a particular data transformation operation on the extracted data

1. Data Cleaning:

* Duplicate Handling Function (handle\_duplicates(a)): accept list of data frames as argument, remove duplicates efficiently and return list of de-duplicated data frames.
* handle\_missing\_values(a): accept de-duplicated data frames, handle null values (remove columns with over 90% nulls & rows with 100% nulls) and return list of cleansed data frames.

1. Data Classification:

* classify\_datatypes(a): this function accepts cleansed df generates 4 new list containing data column names - numerical, date and time, date only and categorical.
* classify\_numeric\_columns(a, b): this function accepts 2 parameters – list of data frames and numerical columns, and further categorize numerical columns into count and measurement columns.

1. Outlier Handling:

* outlier\_detector\_and\_decider\_func(): This uses two other functions (detect\_outlier function and decide\_outlier\_correction function) to generate a dictionary containing information on whether each column has outlier or not and for columns that has outliers, it uses series of logic to decide if the outlier should be removed or adjusted.
* outlier\_corrector\_func(a,b): This accepts 2 parameters – list of cleansed dfs and outlier info dictionary. Outliers in numeric columns are corrected using this function.

1. Skewness and Kurtosis Transformation:

* skewness\_kurtosis\_decider\_func(a,b): This accepts 2 parameters – list of ‘outlier-free dfs’ and measurement column names. Skewness and kurtosis is detected and classed as significant or not based on a threshold.
* transform\_skewness\_kurtosis\_func (a,b): Transformation is applied to reduce skewness and kurtosis on columns with significant skewness and kurtosis. It utilizes box-cox transformation for kurtosis adjustment and log or square root transformation for skewness correction.

1. Data Encoding:

* Categorical columns are encoded for further processing using data\_encoding\_func().

1. Datetime Decomposition:

* Datetime columns are decomposed into date, time, and period\_of\_day components using datetime\_decomposer\_func().

1. Unique ID Generation:

* Unique IDs are generated based on file names and auto-incrementing serial numbers using generate\_unique\_id function.

Load Phase:

1. Data Loading:

* Transformed data is loaded into the destination PostgreSQL database using load\_transformed\_data\_func.
* New columns 'fileimportedby' and 'fileimportdatetime' are added to each DataFrame before loading.

**Conclusion:**

The Fitbit ETL Pipeline streamlines the process of handling Fitbit data, enabling users to extract, transform, and load data efficiently for further analysis and decision-making purposes. It starts with input gathering, followed by data extraction, transformation, and loading phases.

Each phase performs specific tasks to clean, classify, transform, and load the data efficiently. Optimized functions ensure improved performance, readability, and efficiency throughout the pipeline. The pipeline ensures that the extracted data is transformed into a suitable format for analysis or modeling and loaded into the destination database for further use.