**Project**: ETL Tool with Front-End File Upload and Basic Transformation Features

**Delivery Time**: 3 Days

**Overview**:

Our project is an Extract, Transform, Load (ETL) tool that allows users to upload CSV files, perform basic transformations on the data, and save transformation rules for future use. The application consists of a front-end interface built with **React** and **React Query**, and a back-end API built with Django and Django Rest Framework.

The front-end interface allows users to drag and drop CSV files onto the application, which are then displayed in a table format. Users can then drag and drop transformation blocks onto the table to perform basic transformations on the data, such as addition, subtraction, and multiplication. Users can also add or edit transformation rules, and save the rules for future use. The transformed data will be displayed in a table as transformations are being applied.

The back-end API is responsible for handling file uploads, performing transformations on the data, and storing the data and transformation rules in a database. The API is built using **Django** and **Django REST Framework**, and uses **SQL Alchemy** and **Pandas** to perform the data transformations.

Overall, the ETL tool provides a user-friendly interface for uploading, transforming, and analyzing CSV data.

**Requirements**:

* **Front-End Interface:**

Set up a React project.

Use a library such as Material UI or Bootstrap to build the user interface.

Use a drag-and-drop library to implement the drag-and-drop functionality for files and transformation blocks.

Allow users to drag and drop CSV files onto the interface to upload them.

Allow users to drag and drop basic transformation operations (e.g., addition, subtraction, multiplication) onto the data.

Display the transformed data in a table on the interface.

Allow users to save the transformations as rules to be applied to future files.

* **Back-End API:**

Use Django to build a RESTful API to communicate with the front-end interface.

Store the uploaded CSV files in a database.

Store the transformations as rules in the database.

Retrieve the CSV files and rules from the database as needed.

**Additional Requirements:**

Use Python 3.x and the latest versions of Django and Django REST Framework.

Ensure your code is well-structured, easy to read, and follows PEP 8 guidelines.

Handle error cases and return appropriate responses (e.g., 404 Not Found, 400 Bad Request).

**Bonus points:**

1. Include clear and concise documentation in your code to explain your thought process and implementation choices.
2. Add a transformation function to detect anomalies, duplicated and missing values and flag them to the user.
3. Show to transformed data as Charts.
4. Implement more complex transformation operations (e.g., aggregation, filtering, grouping).
5. Implement user authentication and authorization to restrict access to the tool.
6. Include unit tests for your API using Django's testing framework.

**Deliverables**:

Provide a GitHub repository with your project code and documentation.

Include instructions on how to run your project locally for testing and demonstration purposes.