ETL Process for University Data Extraction

The ETL-Python project is designed to facilitate the extraction, transformation, and loading (ETL) of data from various sources into a structured format suitable for analysis and reporting. This documentation provides a comprehensive overview of the project, including its purpose, architecture, setup instructions, usage guidelines, and contribution process.

Table of Contents

1.Project Purpose

The primary goal of the ETL-Python project is to automate data handling, enabling users to efficiently extract data from various sources, transform it into a desired format, and load it into a database or data warehouse. This project benefits data analysts, data scientists, and business intelligence professionals who require clean and structured data for their analyses.

2.Architecture

The architecture of the project consists of three main components:

1. **Extract**: Retrieves data from the Hipolabs Universities API.
2. **Transform**: Filters and restructures the data to focus on universities located in California.
3. **Load**: Saves the transformed data into a SQLite database for easy querying and analysis.

The project also includes a logging mechanism to track the progress and any issues that arise during the ETL process.

3.Setup Instructions

To set up the project, follow these steps:

1. **Clone the Repository**:

```
bash
```

Insert CodeEditCopy code

```
1git clone <repository-url>
2cd <repository-directory>
```

2. **Install Required Packages**: Ensure you have Python installed, then install the required packages using pip:

```
bash
```

Insert CodeEditCopy code

```
1pip install requests pandas sqlalchemy
```

3. **Run the ETL Process**: Execute the script to run the ETL process:

```
bash
```

Insert CodeEditCopy code

```
1python etl_script.py
```

4.Usage

The ETL process can be executed by running the main script. The script will:

- Extract university data from the API.
- Transform the data to filter for California universities.
- Load the transformed data into a SQLite database named `my_lite_store.db`.

After running the script, you can query the database to access the loaded data.

5.File Structure

```
Insert CodeEditCopy code1/etl_project
2
3├── etl_script.py          # Main script for the ETL process
4├── etl.log                # Log file for tracking the ETL process
5└── requirements.txt       # List of required packages
```

6.Contributing

Contributions to the project are welcome! If you would like to contribute, please follow these steps:

1. Fork the repository.
2. Create a new branch for your feature or bug fix.
3. Make your changes and commit them.
4. Push your branch to your forked repository.
5. Create a pull request to the main repository.

7.Components of the Code

7.1 Extract

The extraction process retrieves data from the Hipolabs API, which provides a list of universities in the United States. The extraction function handles potential errors and logs the operation's success or failure.

```
python
```

```
1def extract() -> dict:
2    """ This API extracts data from
```

```
3    http://universities.hipolabs.com
4    """
5    API_URL = "http://universities.hipolabs.com/search?country=United+States"
6    try:
7        data = requests.get(API_URL).json()
8        logger.info('Data extracted successfully')
9        return data
10    except Exception as e:
11        logger.error('Error extracting data: ' + str(e))
12        return None
```

7.2 Transform

The transformation process involves filtering the data to include only California universities. It also restructures the data by combining lists of domains and web pages into single strings. The transformation function logs the number of universities extracted and the number filtered.

```
python
```

```
1def transform(data: dict) -> pd.DataFrame:
2    """ Transforms the dataset into desired structure and filters"""
3    try:
4        df = pd.DataFrame(data)
5        logger.info(f"Total Number of universities from API {len(data)}")
6        df = df[df["name"].str.contains("California")]
7        logger.info(f"Number of universities in california {len(df)}")
8        df['domains'] = [','.join(map(str, l)) for l in df['domains']]
9        df['web_pages'] = [','.join(map(str, l)) for l in df['web_pages']]
10        df = df.reset_index(drop=True)
11        logger.info('Data transformed successfully')
12        return df[["domains", "country", "web_pages", "name"]]
13    except Exception as e:
14        logger.error('Error transforming data: ' + str(e))
15        return None
```

7.3 Load

The loading process saves the transformed data into a SQLite database. The function checks for errors during the loading process and logs the outcome.

```
python
```

```
1def load(df: pd.DataFrame) -> None:
2    """ Loads data into a SQLite database"""
3    try:
4        disk_engine = create_engine('sqlite:///my_lite_store.db')
5        df.to_sql('cal_uni', disk_engine, if_exists='replace')
6        logger.info('Data loaded successfully')
7    except Exception as e:
8        logger.error('Error loading data: ' + str(e))
```

7.4 Logging

The project includes a logging mechanism to track the progress and any issues that arise during the ETL process. Logs are written to both a file (`etl.log`) and the console.

```
python
```

```
1# Create a logger
2logger = logging.getLogger(__name__)
3logger.setLevel(logging.INFO)
4
5# Create a file handler and a stream handler
6file_handler = logging.FileHandler('etl.log')
7stream_handler = logging.StreamHandler()
8
9# Create a formatter and add it to the handlers
10formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
11file_handler.setFormatter(formatter)
12stream_handler.setFormatter(formatter)
13
14# Add the handlers to the logger
15logger.addHandler(file_handler)
16logger.addHandler(stream_handler)
```

8.Conclusion

This ETL process successfully extracts, transforms, and loads university data into a SQLite database. Based on the loaded data, the project can be extended to include additional features such as data analysis or visualization.

9. Future Enhancements

- Implement a scheduling mechanism to run the ETL process periodically.
- Add more filtering options to extract universities based on different criteria.
- Enhance error handling and logging for better monitoring and debugging.