Aim: Every regional load dispatch center in India comes out with generation and load numbers regularly. The format in which the data is released depends upon each center. Aim of this task is to extract date on this page and store it in the database. Database to be used is MySQL/ MariaDB and scripting language to be used should be Python or PHP.

Assumptions:

- Structure of the web page does not change (HTML markup).
- Total number of columns will not exceed the current total of 7 columns.
- Total rows of data is small enough so as to fit into the RAM of hosting server.
- It is legal to scrape data from the website link for personal or commercial use.
- Data is to be inserted only, display of data module is not necessary.
- When there is no data for a particular date, no dummy data is to be inserted.
- Given page does not have dynamically loading content through Ajax.
- Data cannot be inserted partially, wraped in a transaction.
- No use of any Frameworks is specified, thus using core php only.
- User interface is of less importance.

Objectives:

1. Go through the link and create a logical DB structure for the table.
2. Use any of the scripting language mentioned above and extract data for different dates for which data is available.
3. Establish a connection with the designed database using the scripting language.
4. Store the data in the database.
5. Running the script again with same link should only update the entries and not re-insert.
6. Some dates do not have data, e.g. all dates one month or more ago.

Scripting Language: PHP

Database: MariaDB (innoDB Engine)

Approach:

Task given requires a link given to be scraped for specific data. Thus, using *'curl'* fetch website data with a GET request is the basic approach taken. To identify and parse DOM (Document Object Model) elements, a built in PHP class *DOMDocument* is used. To query DOM nodes, another class called *DOMXPath* is used as it provides helpful methods to query DOM nodes.

Basic Algorithm Steps for DOM parsing:

Step 1: Create a curl request to target link.
Step 2: Load returned data into DOMDocument.
Step 3: Query for table in the given page.
Step 4: If table has more than one row (1st row is for column headings), then procees, else show error:

Data not available!
Step 5: If data is available, query all tr elements and loop through each to get desired data. Save it in database. Using transactions here as if we fail inserting all data, then it will not insert partial data. Check if data for current date specified is already inside DB. Update or insert depending on data is present or absent.
Step 6: Display result.
Step 7: Stop.

Note: Website link provided has date format that is incompatible with MySQL/MariaDB format. It is formatted correctly before inserting into database.

Database Design:

Two table are used for the given task! They are *load_date* and *load_values.*

load_date: It stores the date for which load values are stored. Overall structure can be seen through SQL dump file provided!

load_values: It stores the different values of load for a given date. This table contains *load_date_id* as a foreign key referencing *ID* column of *load_date* table.

Above database design is in 2NF. All columns are atomic and all columns as functionally dependent on prime attribute: *ID* in case of *load_date* and *for_date_id* in case of *load_values.*

SQL dump file provides overall in dept structure.

Project Structure:

--Scrape (Root folder)
-----Models
    --TableData.php (Table model)
-----Shared
    --db_config.php (Contains database config items)
--scrape.php (Main file for operation)

Design Considerations:

- To avoid data duplication, tables were split into two.
- Transaction is used to ensure no partial data is inserted into database.
- To save extra data from requests, curl header option is set to 0. Thus, response won't return headers.
- User needs to specify full url to scrape the page as suggested by case study statement.

Procedure to start scraping:

Once application is ready and hosted, open browser and go to the link http://localhost/Scrape/scrape.php. It is assumed that XAMPP server is used.

Input the URL to test and click on Submit button. Wait till you see error or success message about the operation.

Conclusion:

Thus, the given case study has been succesfully performed and its working code is provided as a zip file attached with the email. Case study has been submitted as per given deadine of before 20th January, 2017 10:00 AM.