Hands-on Lab: Acquiring and Processing Information on the World's Largest Banks

Estimated Time: 60 mins

In this project, you will put all the skills acquired throughout the course and your knowledge of basic Python to test. You will work on real-world data and perform the operations of Extraction, Transformation, and Loading (ETL) as required.

Disclaimer:

> Cloud IDE is not a persistent platform, and you will lose your progress every time you restart this lab. We recommend saving a copy of your file on your local machine as a protective measure against data loss.

Project Scenario:

You have been hired as a data engineer by research organization. Your boss has asked you to create a code that can be used to compile the list of the top 10 largest banks in the world ranked by market capitalization in billion USD. Further, the data needs to be transformed and stored in GBP, EUR and INR as well, in accordance with the exchange rate information that has been made available to you as a CSV file. The processed information table is to be saved locally in a CSV format and as a database table.

Your job is to create an automated system to generate this information so that the same can be executed in every financial quarter to prepare the report.

Particulars of the code to be made have been shared below.

Parameter	Value
Code name	`banks_project.py`
Data URL	`https://web.archive.org/web/20230908091635 /https://en.wikipedia.org/wiki/List_of_largest_banks`
Exchange rate CSV path	https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/exchange_rate.csv
Table Attributes (upon Extraction only)	`Name, MC_USD_Billion`
Table Attributes (final)	`Name, MC_USD_Billion, MC_GBP_Billion, MC_EUR_Billion, MC_INR_Billion`
Output CSV Path	`./Largest_banks_data.csv`
Database name	`Banks.db`
Table name	`Largest_banks`
Log file	`code_log.txt`

Project tasks

Task 1:
Write a function `log_progress()` to log the progress of the code at different stages in a file `code_log.txt`. Use the list of log points provided to create log entries as every stage of the code.

Task 2:
Extract the tabular information from the given URL under the heading 'By market capitalization' and save it to a dataframe.
a. Inspect the webpage and identify the position and pattern of the tabular information in the HTML code
b. Write the code for a function `extract()` to perform the required data extraction.
c. Execute a function call to `extract()` to verify the output.

Task 3:
Transform the dataframe by adding columns for Market Capitalization in GBP, EUR and INR, rounded to 2 decimal places, based on the exchange rate information shared as a CSV file.
a. Write the code for a function `transform()` to perform the said task.
b. Execute a function call to `transform()` and verify the output.

Task 4:
Load the transformed dataframe to an output CSV file. Write a function `load_to_csv()`, execute a function call and verify the output.

Task 5:
Load the transformed dataframe to an SQL database server as a table. Write a function `load_to_db()`, execute a function call and verify the output.

Task 6:
Run queries on the database table. Write a function `load_to_db()`, execute a given set of queries and verify the output.

Task 7:
Verify that the log entries have been completed at all stages by checking the contents of the file `code_log.txt`.

Preliminaries: Installing libraries and downloading data

Before building the code, you need to install the required libraries.

The libraries needed for the code are:

`requests` - The library used for accessing the information from the URL.

`bs4` - The library containing the BeautifulSoup function used for webscraping.

`pandas` - The library used for processing the extracted data, storing it in required formats, and communicating with the databases.

`sqlite3` - The library required to create a database server connection.

`numpy` - The library required for the mathematical rounding operations.

`datetime` - The library containing the function datetime used for extracting the timestamp for logging purposes.

Install the required libraries from the terminal window. The command syntax is:

```
python3.11 -m pip install <library_name>
```

Also, download the required exchange rate file using the terminal command:

```
wget     https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/exchange_rate.csv
```

Code Structure

Create the file `banks_project.py` in the path `\home\project\`. Copy and paste the following code structure to the file:

```
# Code for ETL operations on Country-GDP data

# Importing the required libraries

def log_progress(message):
    ''' This function logs the mentioned message of a given stage of the
    code execution to a log file. Function returns nothing'''

def extract(url, table_attribs):
    ''' This function aims to extract the required
    information from the website and save it to a data frame. The
    function returns the data frame for further processing. '''

    return df

def transform(df, csv_path):
    ''' This function accesses the CSV file for exchange rate
    information, and adds three columns to the data frame, each
    containing the transformed version of Market Cap column to
    respective currencies'''

    return df

def load_to_csv(df, output_path):
    ''' This function saves the final data frame as a CSV file in
    the provided path. Function returns nothing.'''

def load_to_db(df, sql_connection, table_name):
    ''' This function saves the final data frame to a database
    table with the provided name. Function returns nothing.'''

def run_query(query_statement, sql_connection):
    ''' This function runs the query on the database table and
    prints the output on the terminal. Function returns nothing. '''

''' Here, you define the required entities and call the relevant
functions in the correct order to complete the project. Note that this
portion is not inside any function.'''
```

At this stage, import the required libraries at the space mentioned in the code structure. Save the file using `Ctrl+S`.

Also, initialize all the known variables as shared in the project scenario.

Task 1: Logging function

Write the function to log the progress of the code, `log_progress()`. The function accepts the message to be logged and enters it to a text file `code_log.txt`.

The format to be used for logging must have the syntax:

```
<time_stamp> : <message>
```

Each log entry must happen in the next line in the text file.

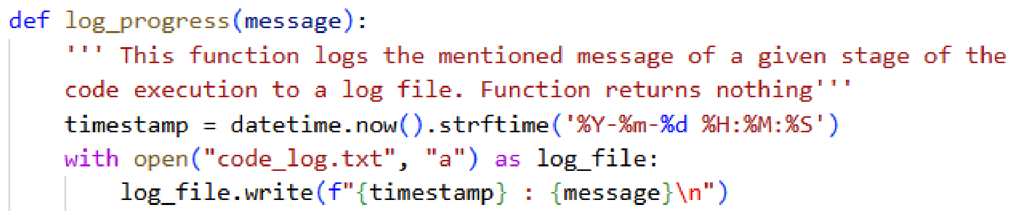
You must associate the correct log entries with each of the executed function calls. Use the following table to note the logging message at the end of each function call that follows.

Task	Log message on completion
Declaring known values	Preliminaries complete. Initiating ETL process
Call extract() function	Data extraction complete. Initiating Transformation process
Call transform() function	Data transformation complete. Initiating Loading process
Call load_to_csv()	Data saved to CSV file
Initiate SQLite3 connection	SQL Connection initiated
Call load_to_db()	Data loaded to Database as a table, Executing queries
Call run_query()	Process Complete
Close SQLite3 connection	Server Connection closed

At this stage, you should now make the first log entry from the table above.

Peer graded assignment prompt:

Take a screenshot of the code, as created for the `log_progress()` function and save it to your local machine as `Task_1_log_function.png`



```python
def log_progress(message):
    ''' This function logs the mentioned message of a given stage of the
    code execution to a log file. Function returns nothing'''
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    with open("code_log.txt", "a") as log_file:
        log_file.write(f"{timestamp} : {message}\n")
```

Task 2 : Extraction of data

Analyze the webpage on the given URL:

```
1. https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List_of_largest_banks
```

Identify the position of the required table under the heading `By market capitalization`. Write the function `extract()` to retrieve the information of the table to a Pandas data frame.

Note: Remember to remove the last character from the `Market Cap` column contents, like, '\n', and typecast the value to float format.

Write a function call for `extract()` and print the returning data frame.

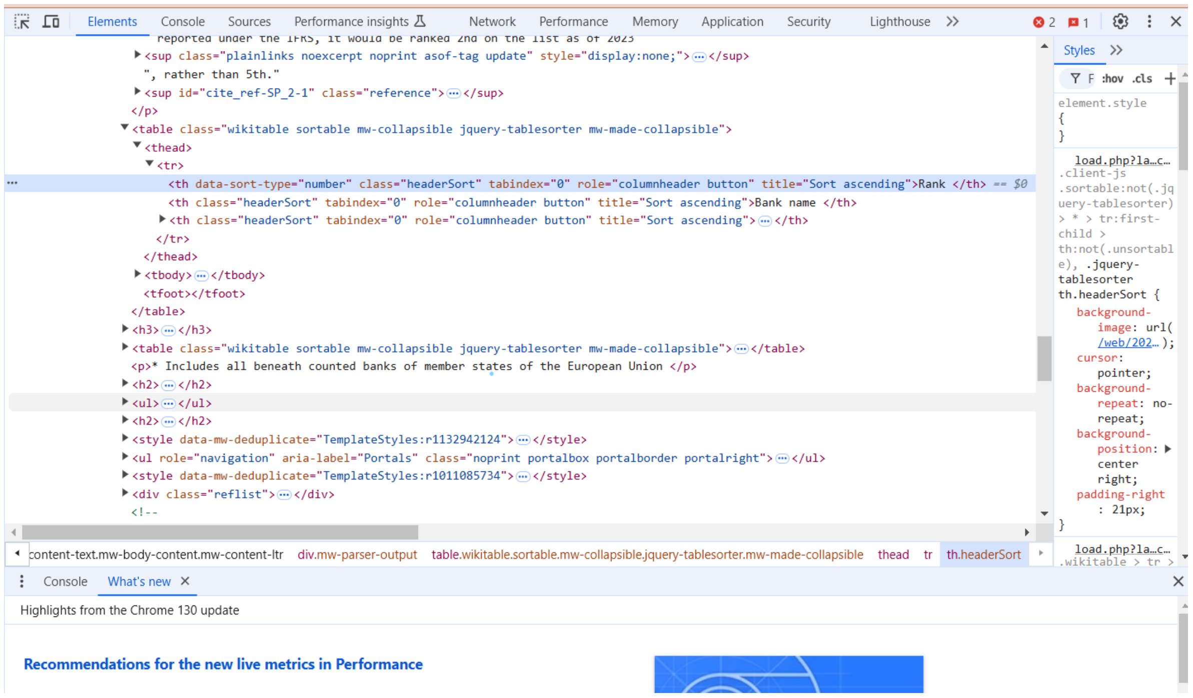
Make the relevant log entry.

Execute the code using the command:

```
1. python3.11 banks_project.py
```

Peer graded assignment prompt:

Take a screenshot of the html code of the table, as obtained by inspecting the webpage. Make sure that the contents of at least the first row of the table, as entered in the HTML code, are completely visible. Save this screenshot to your local machine as `Task_2a_extract.png`



```html
reported under the IFRS, it would be ranked 2nd on the list as of 2023
  <sup class="plainlinks noexcerpt noprint asof-tag update" style="display:none;">…</sup>
  ", rather than 5th."
  <sup id="cite_ref-SP_2-1" class="reference">…</sup>
</p>
<table class="wikitable sortable mw-collapsible jquery-tablesorter mw-made-collapsible">
  <thead>
    <tr>
      <th data-sort-type="number" class="headerSort" tabindex="0" role="columnheader button" title="Sort ascending">Rank </th> == $0
      <th class="headerSort" tabindex="0" role="columnheader button" title="Sort ascending">Bank name </th>
      <th class="headerSort" tabindex="0" role="columnheader button" title="Sort ascending">…</th>
    </tr>
  </thead>
  <tbody>…</tbody>
  <tfoot></tfoot>
</table>
<h3>…</h3>
<table class="wikitable sortable mw-collapsible jquery-tablesorter mw-made-collapsible">…</table>
<p>* Includes all beneath counted banks of member states of the European Union </p>
<h2>…</h2>
<ul>…</ul>
<h2>…</h2>
<style data-mw-deduplicate="TemplateStyles:r1132942124">…</style>
<ul role="navigation" aria-label="Portals" class="noprint portalbox portalborder portalright">…</ul>
<style data-mw-deduplicate="TemplateStyles:r1011085734">…</style>
<div class="reflist">…</div>
<!--
```

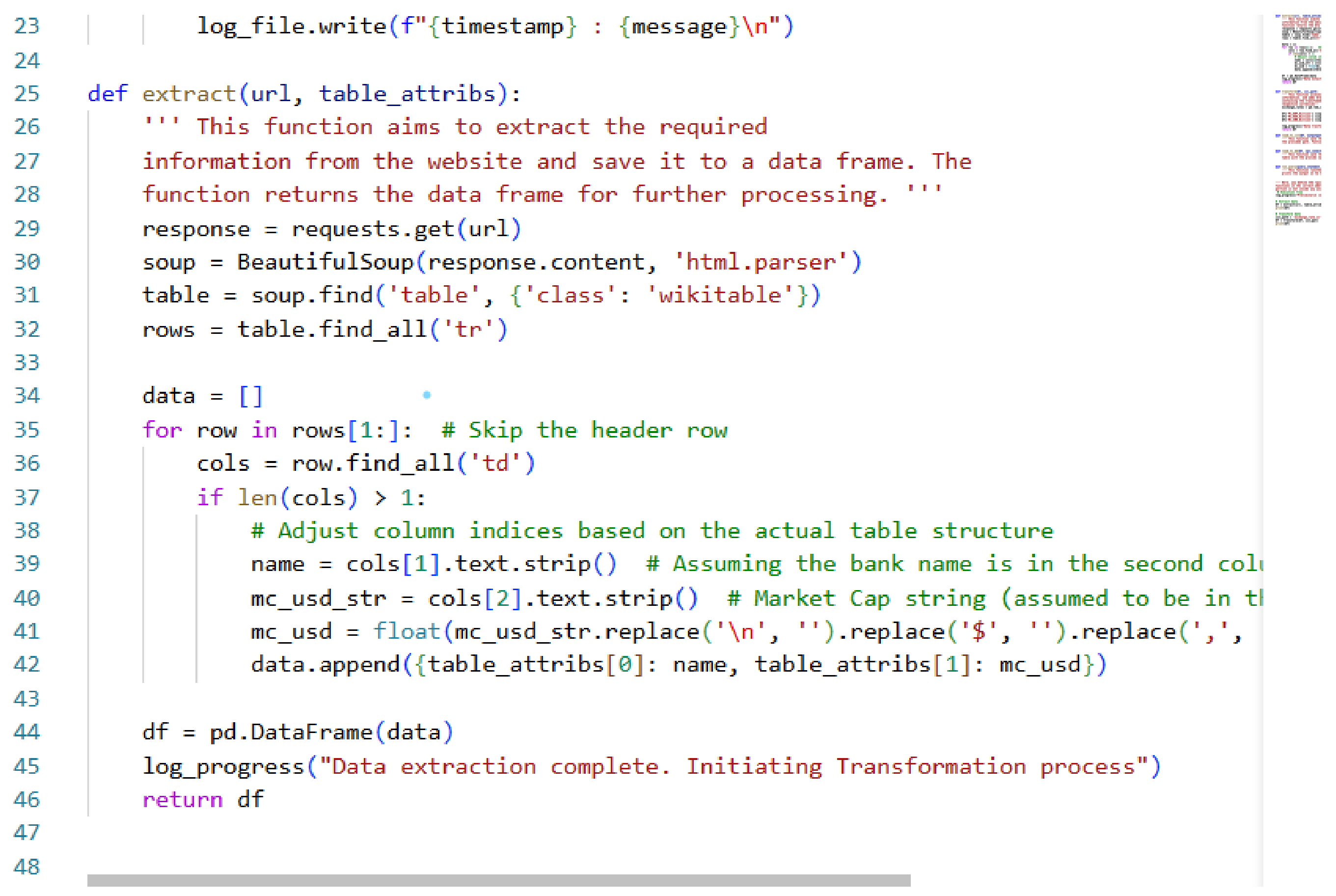
```
content-text.mw-body-content.mw-content-ltr  div.mw-parser-output  table.wikitable.sortable.mw-collapsible.jquery-tablesorter.mw-made-collapsible  thead  tr  th.headerSort
```

```css
element.style {
}
load.php?la…c…
.client-js .sortable:not(.jquery-tablesorter) > * > tr:first-child > th:not(.unsortable), .jquery-tablesorter th.headerSort {
    background-image: url(/web/202…);
    cursor: pointer;
    background-repeat: no-repeat;
    background-position: center right;
    padding-right: 21px;
}
load.php?la…c…
.wikitable > tr >
```

Highlights from the Chrome 130 update

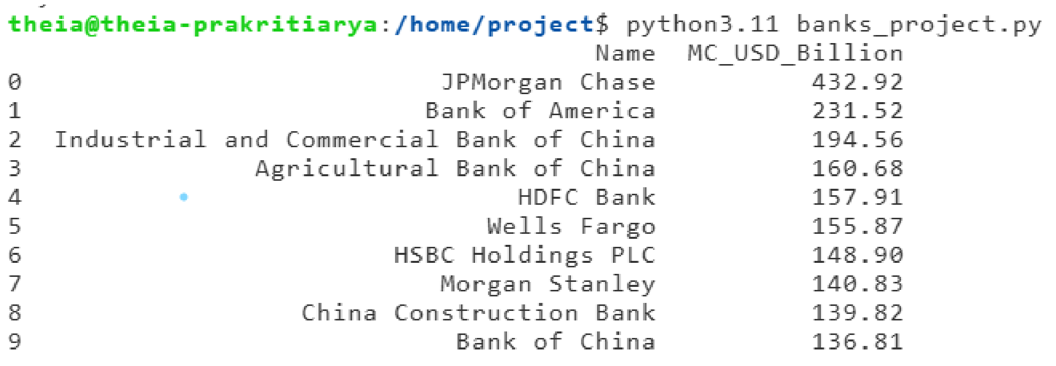
Recommendations for the new live metrics in Performance

Take a screenshot of the code, as created for the `extract()` function and save it to your local machine as `Task_2b_extract.png`.



```
            log_file.write(f"{timestamp} : {message}\n")

def extract(url, table_attribs):
    ''' This function aims to extract the required
    information from the website and save it to a data frame. The
    function returns the data frame for further processing. '''
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'wikitable'})
    rows = table.find_all('tr')

    data = []
    for row in rows[1:]:  # Skip the header row
        cols = row.find_all('td')
        if len(cols) > 1:
            # Adjust column indices based on the actual table structure
            name = cols[1].text.strip()  # Assuming the bank name is in the second colu
            mc_usd_str = cols[2].text.strip()  # Market Cap string (assumed to be in th
            mc_usd = float(mc_usd_str.replace('\n', '').replace('$', '').replace(',',
            data.append({table_attribs[0]: name, table_attribs[1]: mc_usd})

    df = pd.DataFrame(data)
    log_progress("Data extraction complete. Initiating Transformation process")
    return df

```

Take a screenshot of the output, as obtained upon execution in the terminal, and save it to your local machine as `Task_2c_extract.png`



```
theia@theia-prakritiarya:/home/project$ python3.11 banks_project.py
                                      Name  MC_USD_Billion
0                           JPMorgan Chase          432.92
1                          Bank of America          231.52
2  Industrial and Commercial Bank of China          194.56
3               Agricultural Bank of China          160.68
4                                HDFC Bank          157.91
5                              Wells Fargo          155.87
6                        HSBC Holdings PLC          148.90
7                           Morgan Stanley          140.83
8                  China Construction Bank          139.82
9                            Bank of China          136.81
```

Task 3 : Transformation of data

The Transform function needs to perform the following tasks:

1. Read the exchange rate CSV file and convert the contents to a dictionary so that the contents of the first columns are the keys to the dictionary and the contents of the second column are the corresponding values.

2. Add 3 different columns to the dataframe, viz. `MC_GBP_Billion`, `MC_EUR_Billion` and `MC_INR_Billion`, each containing the content of

`MC_USD_Billion` scaled by the corresponding exchange rate factor. Remember to round the resulting data to 2 decimal places.

A sample statement is being provided for adding the `MC_GBP_Billion` column. You can use this to add the other two statements on your own.

```
1. df['MC_GBP_Billion'] = [np.round(x*exchange_rate['GBP'],2) for x in df['MC_USD_Billion']]
```

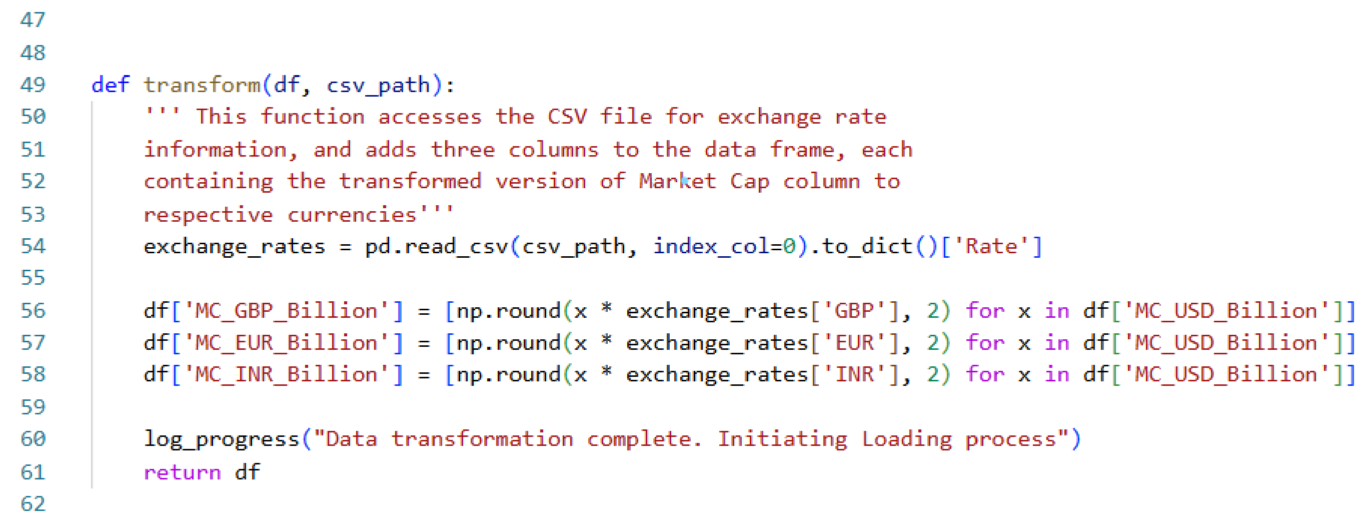
Write the function call for `transform()` and print the contents of the returning data frame. Comment out all previous print statements.

Make the relevant log entry and execute the code.

Print the contents of df['MC_EUR_Billion'][4], which is the market capitalization of the 5th largest bank in billion EUR

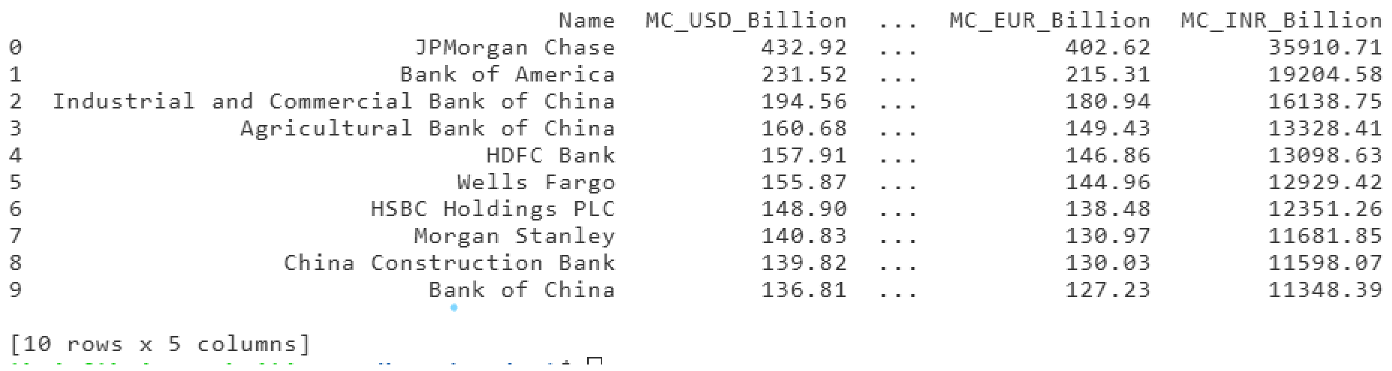
Peer graded assignment prompt:

Take a screenshot of the code, as created for the `transform()` function, and save it to your local machine as `Task_3a_transform.png`.



```
def transform(df, csv_path):
    ''' This function accesses the CSV file for exchange rate
    information, and adds three columns to the data frame, each
    containing the transformed version of Market Cap column to
    respective currencies'''
    exchange_rates = pd.read_csv(csv_path, index_col=0).to_dict()['Rate']

    df['MC_GBP_Billion'] = [np.round(x * exchange_rates['GBP'], 2) for x in df['MC_USD_Billion']]
    df['MC_EUR_Billion'] = [np.round(x * exchange_rates['EUR'], 2) for x in df['MC_USD_Billion']]
    df['MC_INR_Billion'] = [np.round(x * exchange_rates['INR'], 2) for x in df['MC_USD_Billion']]

    log_progress("Data transformation complete. Initiating Loading process")
    return df
```

Take a snapshot of the output and save it as `Task_3b_tranform.png`.



```
                                   Name  MC_USD_Billion  ...  MC_EUR_Billion  MC_INR_Billion
0                        JPMorgan Chase          432.92  ...          402.62        35910.71
1                       Bank of America          231.52  ...          215.31        19204.58
2  Industrial and Commercial Bank of China       194.56  ...          180.94        16138.75
3            Agricultural Bank of China          160.68  ...          149.43        13328.41
4                             HDFC Bank          157.91  ...          146.86        13098.63
5                           Wells Fargo          155.87  ...          144.96        12929.42
6                     HSBC Holdings PLC          148.90  ...          138.48        12351.26
7                        Morgan Stanley          140.83  ...          130.97        11681.85
8               China Construction Bank          139.82  ...          130.03        11598.07
9                         Bank of China          136.81  ...          127.23        11348.39

[10 rows x 5 columns]
```

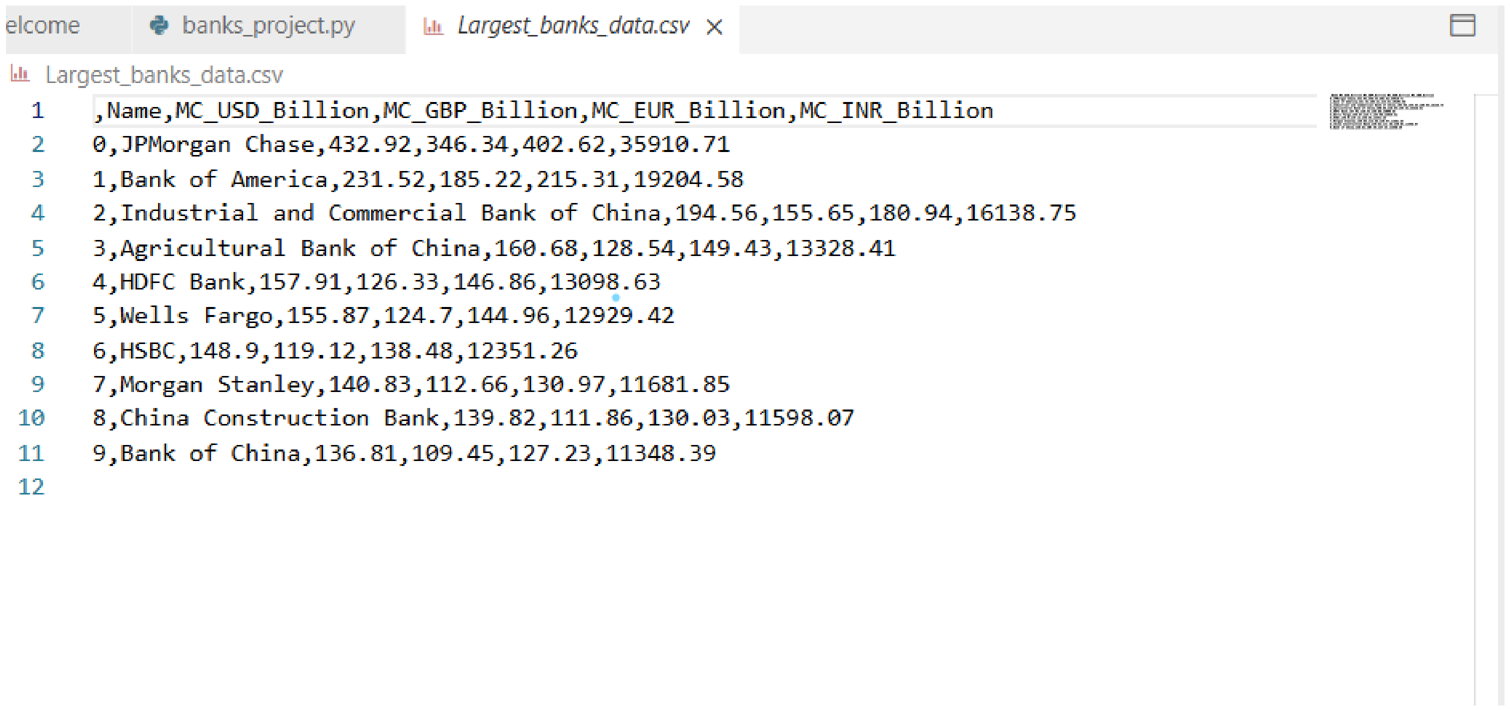
Task 4: Loading to CSV

Write the function to load the transformed data frame to a CSV file, like `load_to_csv()`, in the path mentioned in the project scenario.

Make the relevant log entry.

Peer graded assignment prompt:

Double-click the created CSV file in the `Explorer` tab on the left ribbon of the programming pane in Cloud IDE. Note that its contents are displayed on the editor screen. Take a snapshot of this screen and save it as `Task_4_CSV.png`.



```
,Name,MC_USD_Billion,MC_GBP_Billion,MC_EUR_Billion,MC_INR_Billion
0,JPMorgan Chase,432.92,346.34,402.62,35910.71
1,Bank of America,231.52,185.22,215.31,19204.58
2,Industrial and Commercial Bank of China,194.56,155.65,180.94,16138.75
3,Agricultural Bank of China,160.68,128.54,149.43,13328.41
4,HDFC Bank,157.91,126.33,146.86,13098.63
5,Wells Fargo,155.87,124.7,144.96,12929.42
6,HSBC,148.9,119.12,138.48,12351.26
7,Morgan Stanley,140.83,112.66,130.97,11681.85
8,China Construction Bank,139.82,111.86,130.03,11598.07
9,Bank of China,136.81,109.45,127.23,11348.39
```

Task 5: Loading to Database

Write the function to load the transformed data frame to an SQL database, like, `load_to_db()`. Use the database and table names as mentioned in the project scenario.

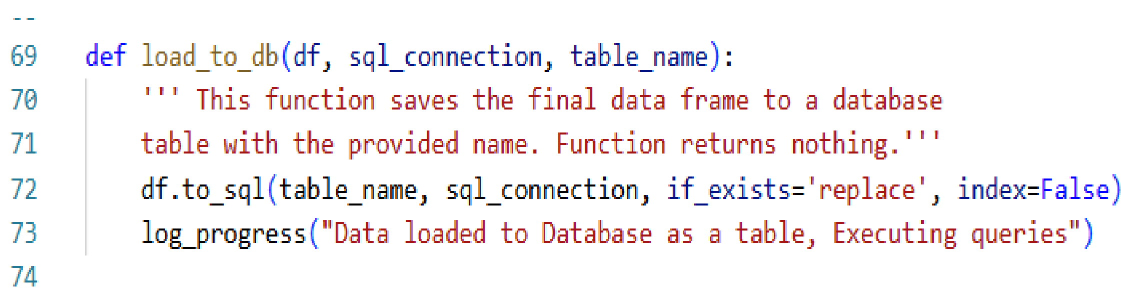
Before calling this function, initiate the connection to the SQLite3 database server with the name `Banks.db`. Pass this connection object, along with the required table name `Largest_banks` and the transformed data frame, to the `load_to_db()` function in the function call.

Make the relevant log entry.

Upon successful function call, you will have loaded the contents of the table with the required data and the file `Banks.db` will be visible in the `Explorer` tab of the IDE under the `project` folder.

Peer graded assignment prompt:

Take a single screenshot of the code, as created for `load_to_csv()` and `load_to_db()` functions, and save it to your local machine as `Task_4_5_save_file.png`.



```
def load_to_db(df, sql_connection, table_name):
    ''' This function saves the final data frame to a database
    table with the provided name. Function returns nothing.'''
    df.to_sql(table_name, sql_connection, if_exists='replace', index=False)
    log_progress("Data loaded to Database as a table, Executing queries")
```

Task 6: Function to Run queries on Database

Write the function `run_queries()` that accepts the query statement, and the SQLite3 Connection object, and generates the output of the query. The query statement should be printed along with the query output.

Execute 3 function calls using the queries as mentioned below.

1. Print the contents of the entire table

Query statement:

```
1. SELECT * FROM Largest_banks
```

2. Print the average market capitalization of all the banks in Billion USD.

Query statement:

```
1. SELECT AVG(MC_GBP_Billion) FROM Largest_banks
```

3. Print only the names of the top 5 banks

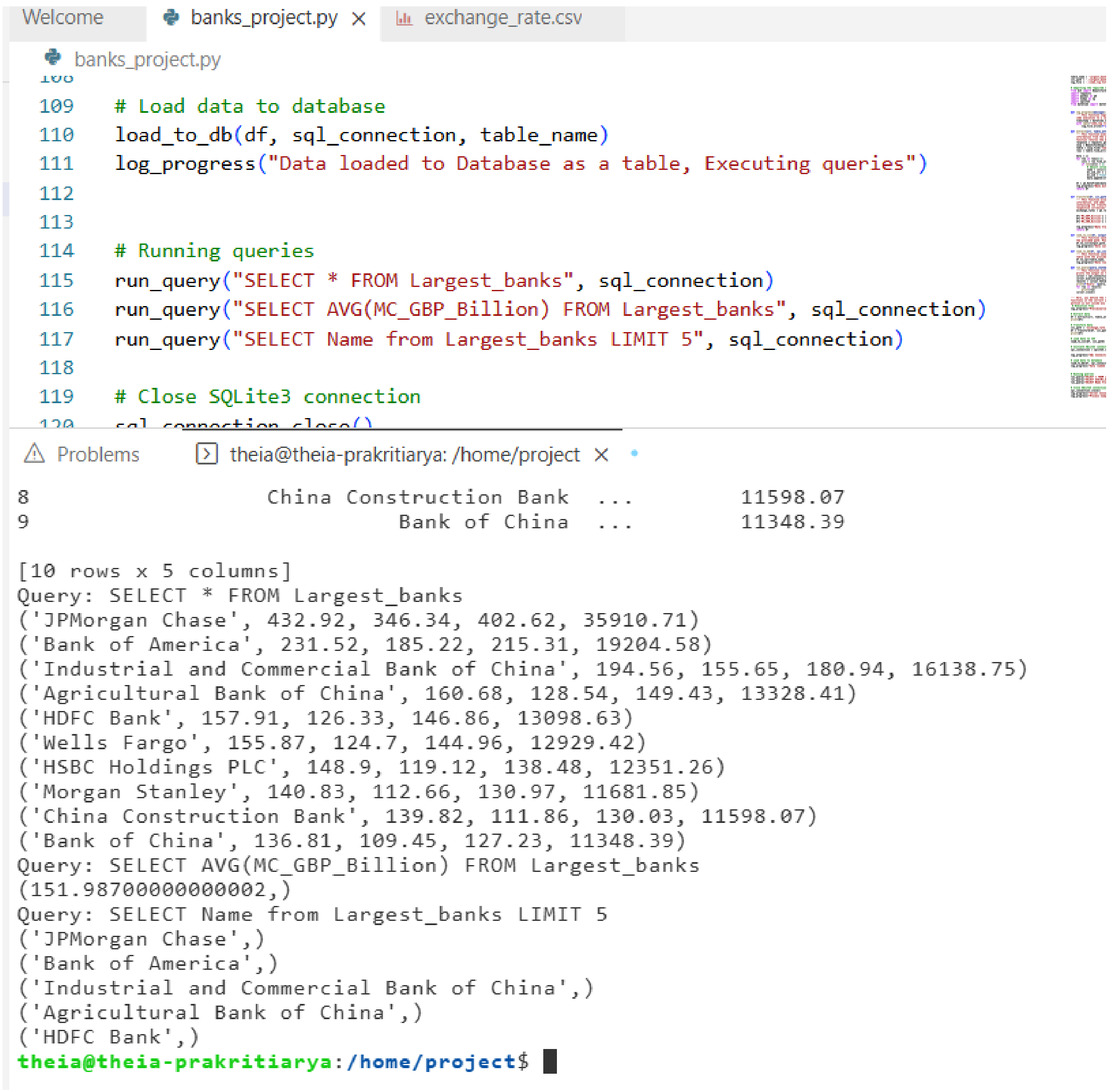
Query statement:

```
1. SELECT Name from Largest_banks LIMIT 5
```

Make the relevant log entry.

Peer graded assignment prompt:

Take the snapshot of the output and save it as `Task_6_SQL.png`. Please adjust the size of the terminal prompt in order to take a single screenshot that captures all three outputs together.



```
    # Load data to database
    load_to_db(df, sql_connection, table_name)
    log_progress("Data loaded to Database as a table, Executing queries")


    # Running queries
    run_query("SELECT * FROM Largest_banks", sql_connection)
    run_query("SELECT AVG(MC_GBP_Billion) FROM Largest_banks", sql_connection)
    run_query("SELECT Name from Largest_banks LIMIT 5", sql_connection)

    # Close SQLite3 connection
```

```
8                  China Construction Bank  ...        11598.07
9                            Bank of China  ...        11348.39

[10 rows x 5 columns]
Query: SELECT * FROM Largest_banks
('JPMorgan Chase', 432.92, 346.34, 402.62, 35910.71)
('Bank of America', 231.52, 185.22, 215.31, 19204.58)
('Industrial and Commercial Bank of China', 194.56, 155.65, 180.94, 16138.75)
('Agricultural Bank of China', 160.68, 128.54, 149.43, 13328.41)
('HDFC Bank', 157.91, 126.33, 146.86, 13098.63)
('Wells Fargo', 155.87, 124.7, 144.96, 12929.42)
('HSBC Holdings PLC', 148.9, 119.12, 138.48, 12351.26)
('Morgan Stanley', 140.83, 112.66, 130.97, 11681.85)
('China Construction Bank', 139.82, 111.86, 130.03, 11598.07)
('Bank of China', 136.81, 109.45, 127.23, 11348.39)
Query: SELECT AVG(MC_GBP_Billion) FROM Largest_banks
(151.98700000000002,)
Query: SELECT Name from Largest_banks LIMIT 5
('JPMorgan Chase',)
('Bank of America',)
('Industrial and Commercial Bank of China',)
('Agricultural Bank of China',)
('HDFC Bank',)
theia@theia-prakritiarya:/home/project$
```

Task 7: Verify log entries

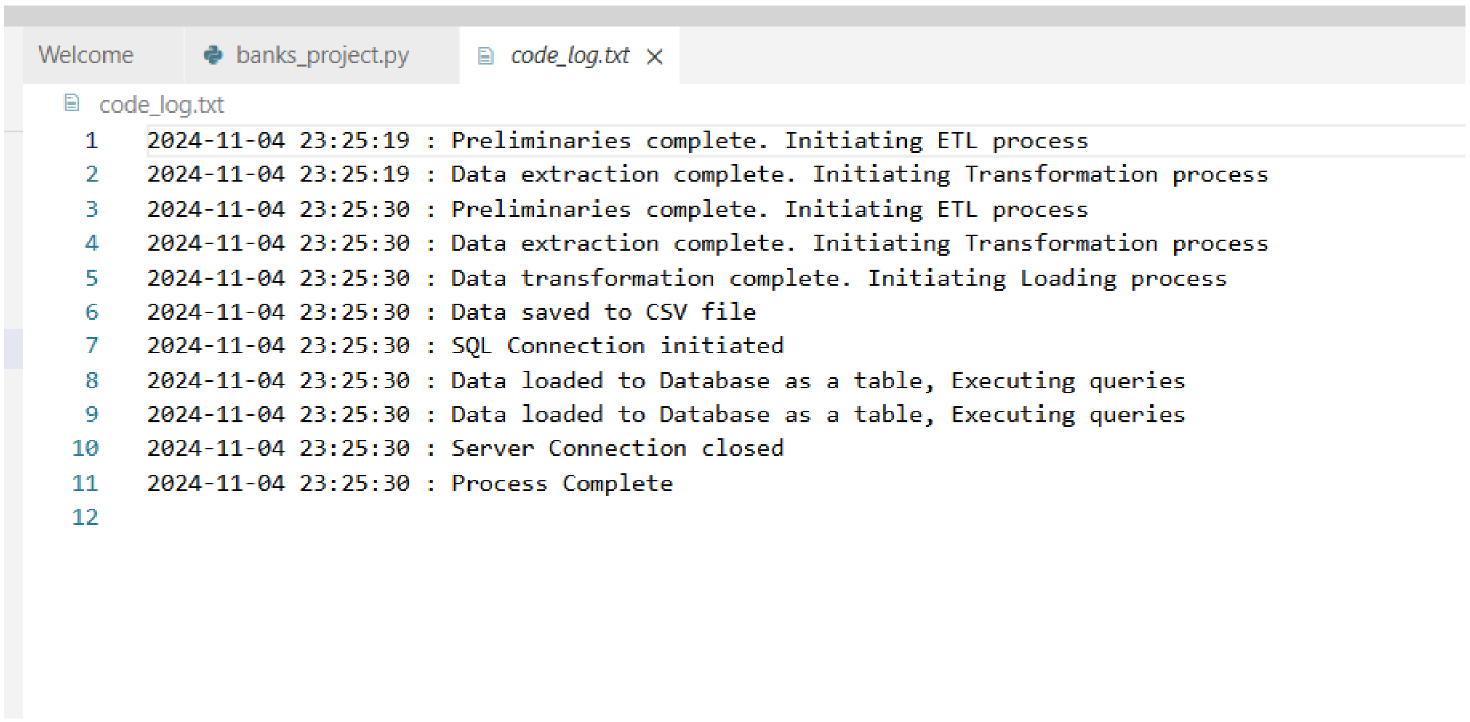
After updating all the `log_progress()` function calls, you have to run the code for a final execution. However, you will first have to remove the `code_log.txt` file, that would have been created and updated throughout the multiple executions of the code in this lab. You may remove the file using the following command on a terminal.

```
1. rm code_log.txt
```

Once the existing file is removed, now run the final execution. Upon successful completion of execution, open the `code_log.txt` file by clicking on it in the `Explorer` tab of the toolbar on left side of the programming pane of the IDE, under the `project` folder. You should see all the relevant entries made in the text file in relation to the stages of code execution.

Peer graded assignment prompt:

Take the snapshot of the file contents and save it as `Task_7_log_content.png`.



```
2024-11-04 23:25:19 : Preliminaries complete. Initiating ETL process
2024-11-04 23:25:19 : Data extraction complete. Initiating Transformation process
2024-11-04 23:25:30 : Preliminaries complete. Initiating ETL process
2024-11-04 23:25:30 : Data extraction complete. Initiating Transformation process
2024-11-04 23:25:30 : Data transformation complete. Initiating Loading process
2024-11-04 23:25:30 : Data saved to CSV file
2024-11-04 23:25:30 : SQL Connection initiated
2024-11-04 23:25:30 : Data loaded to Database as a table, Executing queries
2024-11-04 23:25:30 : Data loaded to Database as a table, Executing queries
2024-11-04 23:25:30 : Server Connection closed
2024-11-04 23:25:30 : Process Complete
```

Conclusion

Congratulations on completing this project!

With this, you are now trained to perform ETL operations on real-world data and make the processed information available for further use in different formats.

You should now be able to:

- Use Webscraping techniques to extract information from any website as per requirement.
- Use Pandas data frames and dictionaries to transform data as per requirement.
- Load the processed information to CSV files and as Database tables
- Query the database tables using SQLite3 and pandas libraries
- Log the progress of the code properly

Author(s)

Abhishek Gagneja