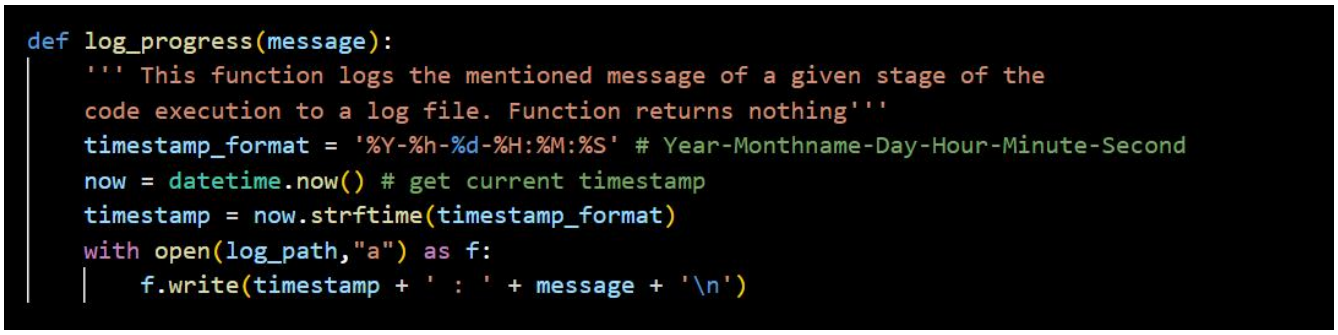
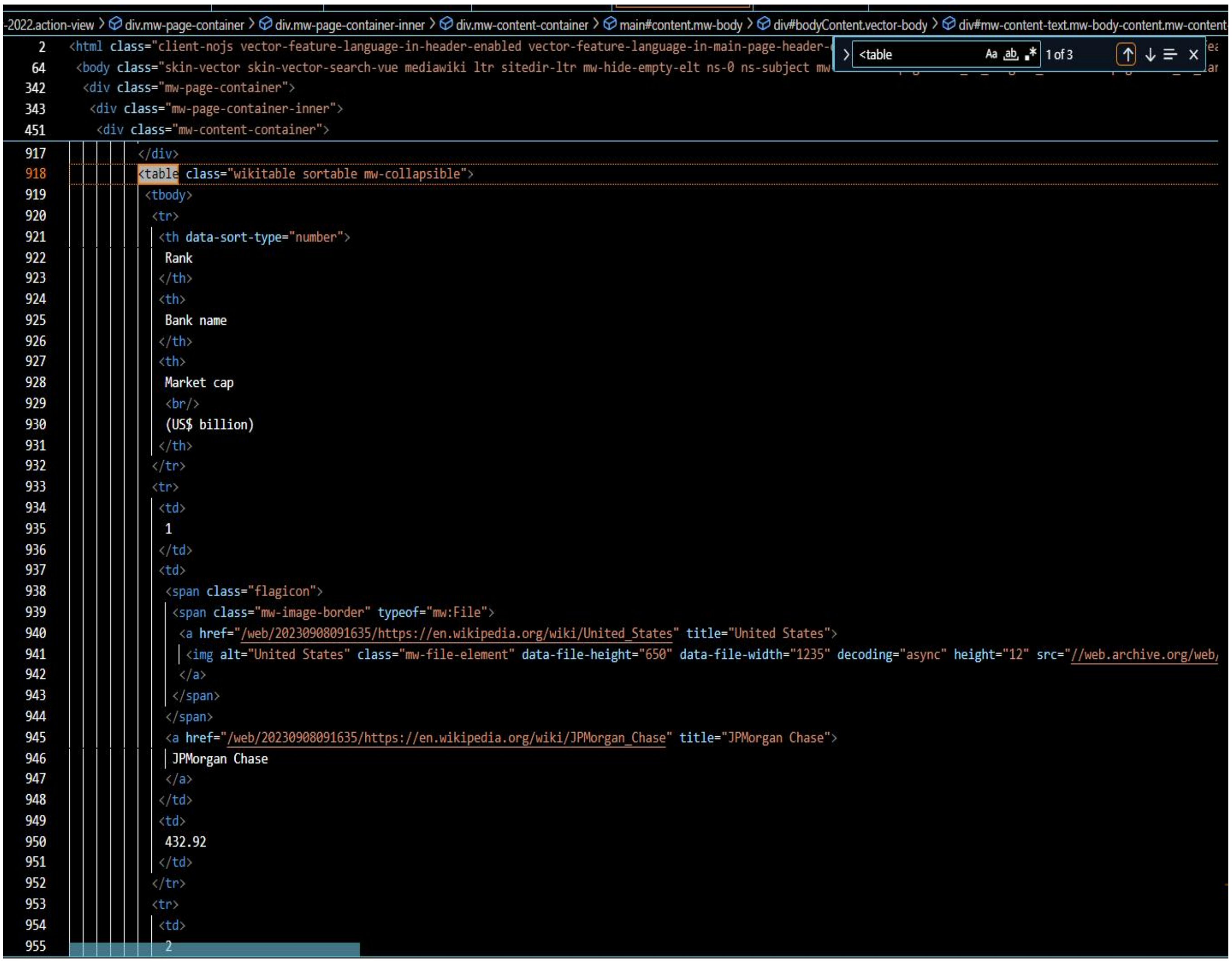
Final Graded Project Submission

1. Log Progress code snippet:



```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_format = '%Y-%h-%d-%H:%M:%S' # Year-Monthname-Day-Hour-Minute-Second
    now = datetime.now() # get current timestamp
    timestamp = now.strftime(timestamp_format)
    with open(log_path,"a") as f:
        f.write(timestamp + ' : ' + message + '\n')
```

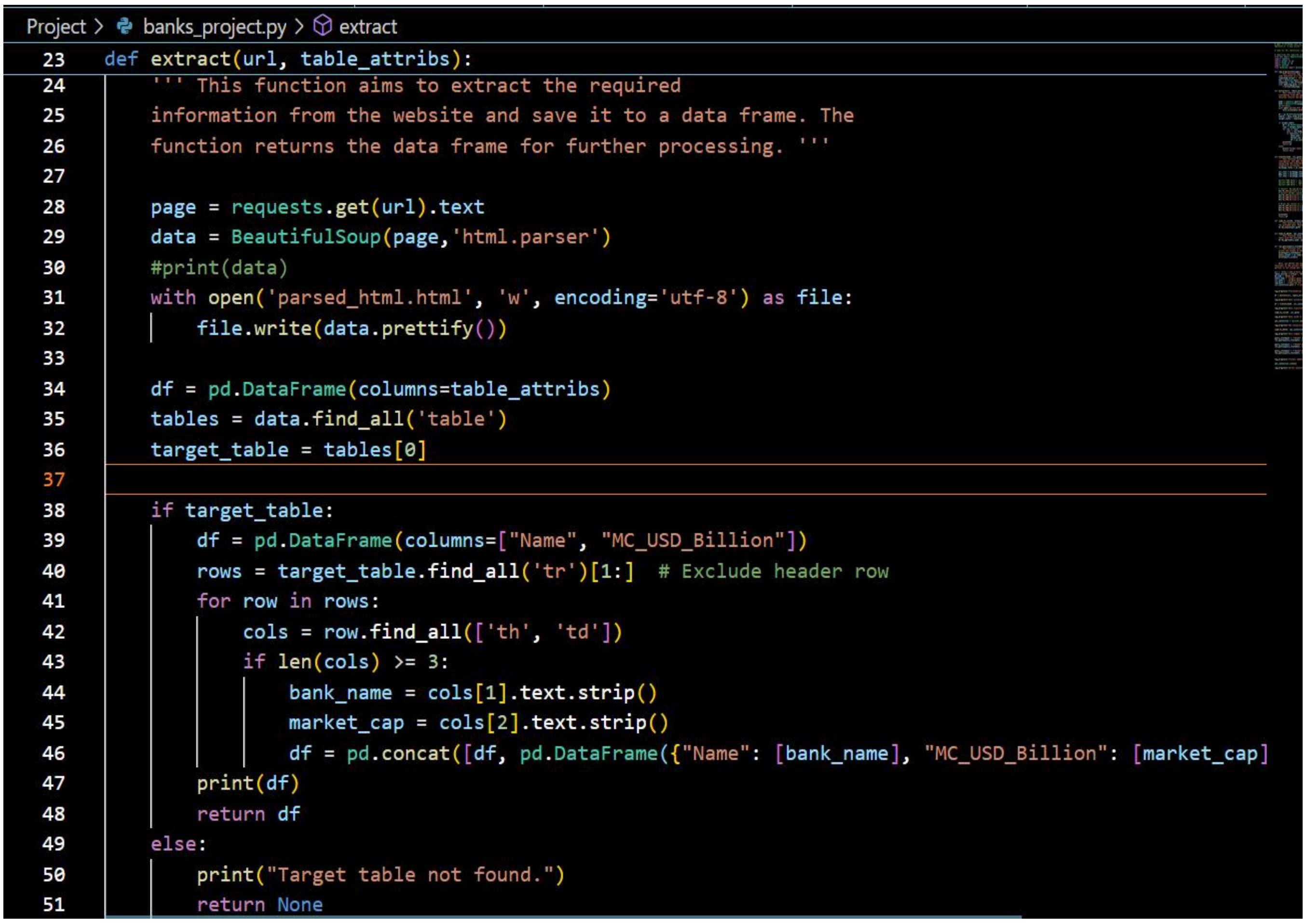
2. Parsed HTML with at-least one of the row to be extracted:



```
-2022.action-view > div.mw-page-container > div.mw-page-container-inner > div.mw-content-container > main#content.mw-body > div#bodyContent.vector-body > div#mw-content-text.mw-body-content.mw-content
```

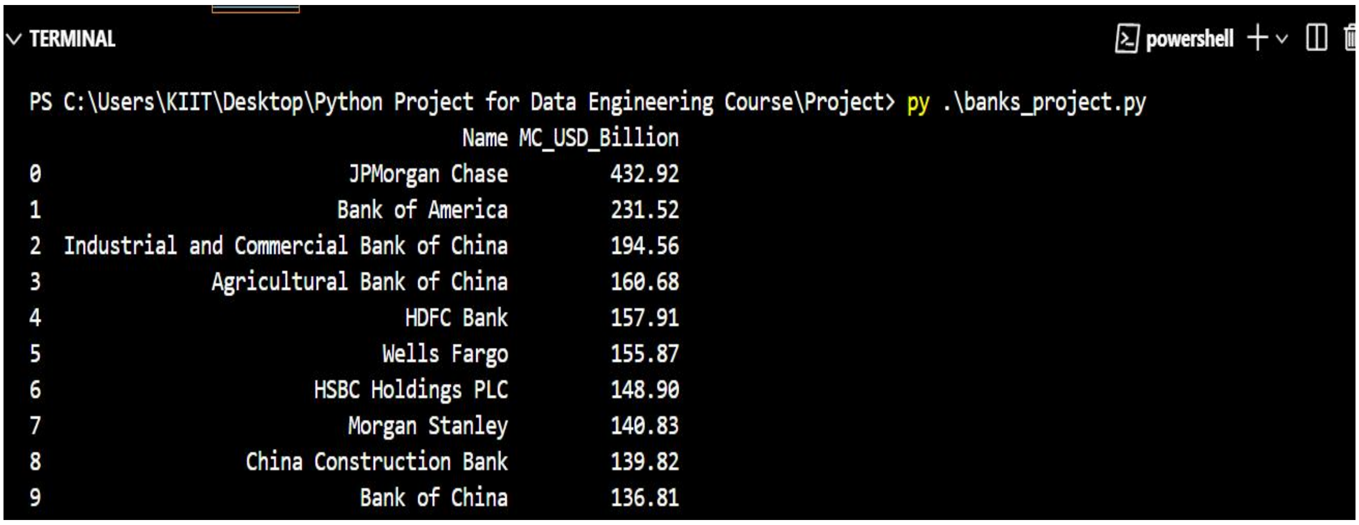
```html
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-
 <body class="skin-vector skin-vector-search-vue mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject mw
  <div class="mw-page-container">
   <div class="mw-page-container-inner">
    <div class="mw-content-container">
          </div>
          <table class="wikitable sortable mw-collapsible">
           <tbody>
            <tr>
             <th data-sort-type="number">
              Rank
             </th>
             <th>
              Bank name
             </th>
             <th>
              Market cap
              <br/>
              (US$ billion)
             </th>
            </tr>
            <tr>
             <td>
              1
             </td>
             <td>
              <span class="flagicon">
               <span class="mw-image-border" typeof="mw:File">
                <a href="/web/20230908091635/https://en.wikipedia.org/wiki/United_States" title="United States">
                 <img alt="United States" class="mw-file-element" data-file-height="650" data-file-width="1235" decoding="async" height="12" src="//web.archive.org/web,
                </a>
               </span>
              </span>
              <a href="/web/20230908091635/https://en.wikipedia.org/wiki/JPMorgan_Chase" title="JPMorgan Chase">
               JPMorgan Chase
              </a>
             </td>
             <td>
              432.92
             </td>
            </tr>
            <tr>
             <td>
              2
```

3. Extract function code snippet:



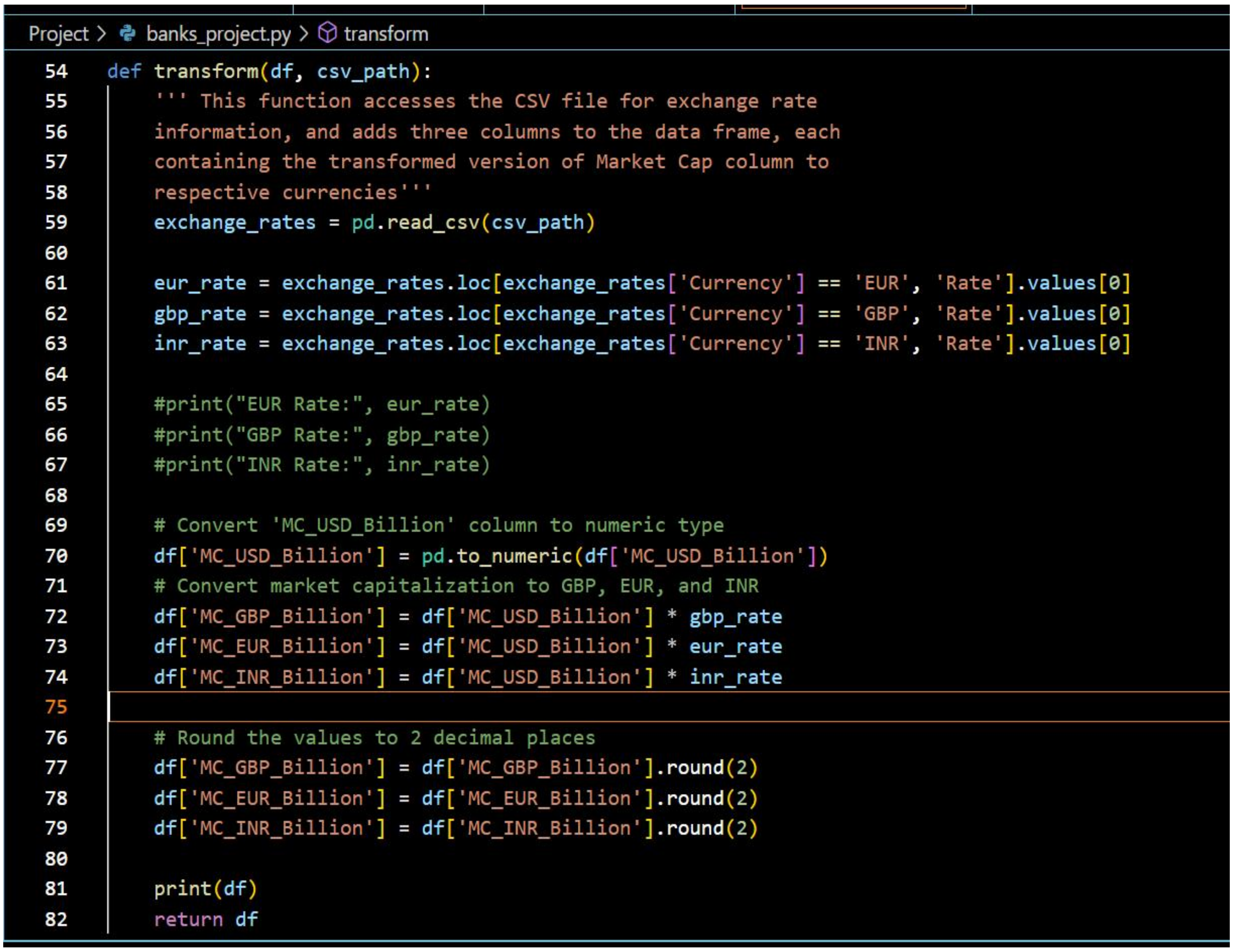
```python
def extract(url, table_attribs):
    ''' This function aims to extract the required
    information from the website and save it to a data frame. The
    function returns the data frame for further processing. '''

    page = requests.get(url).text
    data = BeautifulSoup(page,'html.parser')
    #print(data)
    with open('parsed_html.html', 'w', encoding='utf-8') as file:
        file.write(data.prettify())

    df = pd.DataFrame(columns=table_attribs)
    tables = data.find_all('table')
    target_table = tables[0]

    if target_table:
        df = pd.DataFrame(columns=["Name", "MC_USD_Billion"])
        rows = target_table.find_all('tr')[1:]  # Exclude header row
        for row in rows:
            cols = row.find_all(['th', 'td'])
            if len(cols) >= 3:
                bank_name = cols[1].text.strip()
                market_cap = cols[2].text.strip()
                df = pd.concat([df, pd.DataFrame({"Name": [bank_name], "MC_USD_Billion": [market_cap]
        print(df)
        return df
    else:
        print("Target table not found.")
        return None
```

4. Output after executing extract function:



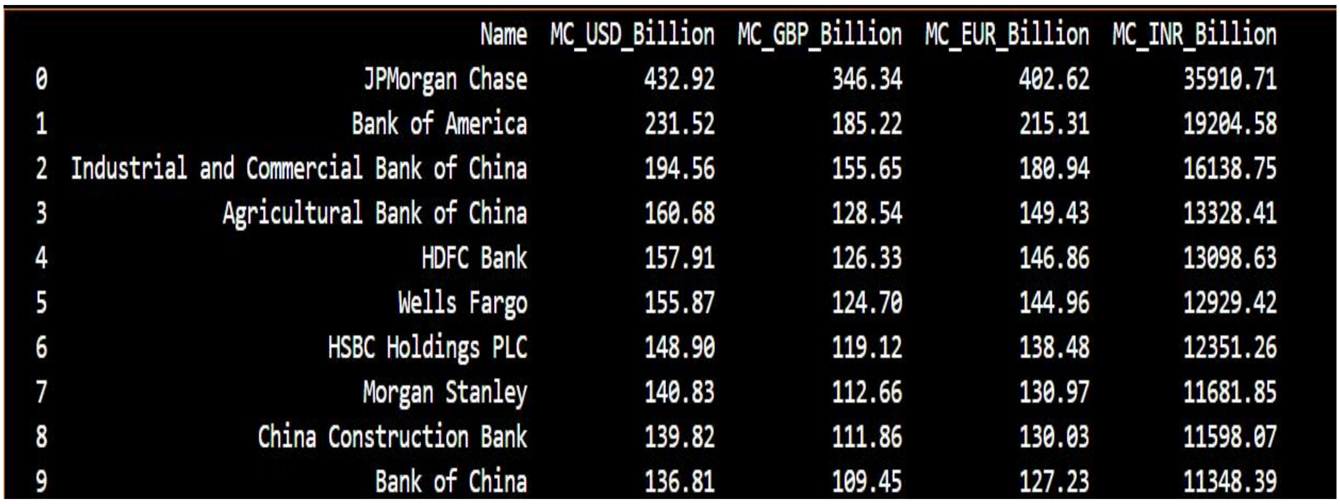
```
PS C:\Users\KIIT\Desktop\Python Project for Data Engineering Course\Project> py .\banks_project.py
                                    Name MC_USD_Billion
0                         JPMorgan Chase         432.92
1                        Bank of America         231.52
2  Industrial and Commercial Bank 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
```

5. Transform function code snippet:



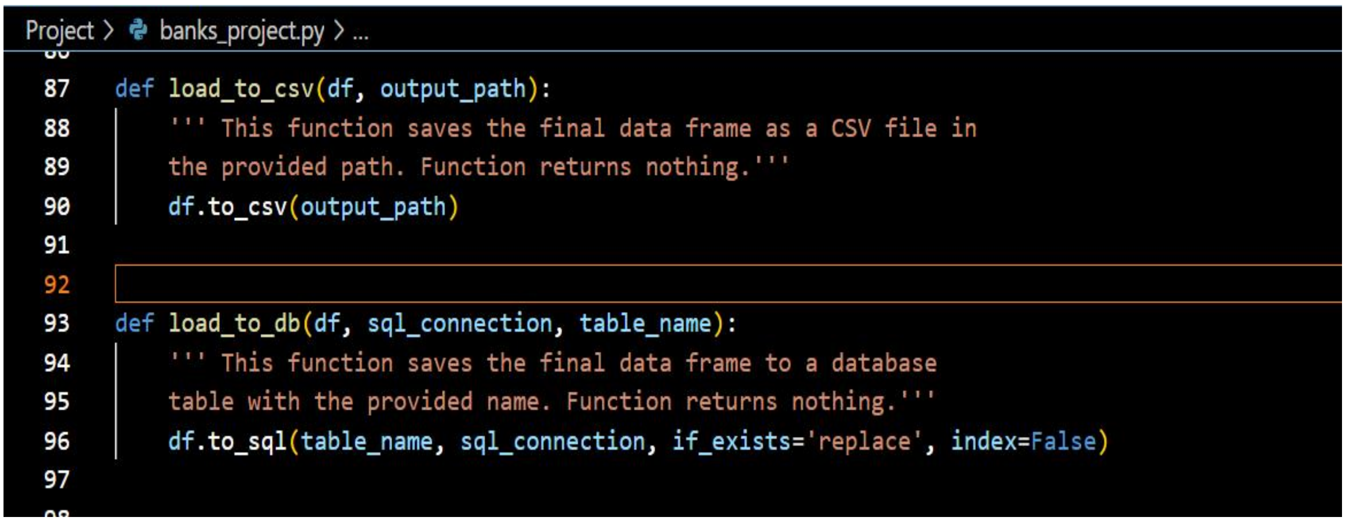
```python
def transform(df, csv_path):
    ''' This function accesses the CSV file for exchange rate
    information, and adds three columns to the data frame, each
    containing the transformed version of Market Cap column to
    respective currencies'''
    exchange_rates = pd.read_csv(csv_path)

    eur_rate = exchange_rates.loc[exchange_rates['Currency'] == 'EUR', 'Rate'].values[0]
    gbp_rate = exchange_rates.loc[exchange_rates['Currency'] == 'GBP', 'Rate'].values[0]
    inr_rate = exchange_rates.loc[exchange_rates['Currency'] == 'INR', 'Rate'].values[0]

    #print("EUR Rate:", eur_rate)
    #print("GBP Rate:", gbp_rate)
    #print("INR Rate:", inr_rate)

    # Convert 'MC_USD_Billion' column to numeric type
    df['MC_USD_Billion'] = pd.to_numeric(df['MC_USD_Billion'])
    # Convert market capitalization to GBP, EUR, and INR
    df['MC_GBP_Billion'] = df['MC_USD_Billion'] * gbp_rate
    df['MC_EUR_Billion'] = df['MC_USD_Billion'] * eur_rate
    df['MC_INR_Billion'] = df['MC_USD_Billion'] * inr_rate

    # Round the values to 2 decimal places
    df['MC_GBP_Billion'] = df['MC_GBP_Billion'].round(2)
    df['MC_EUR_Billion'] = df['MC_EUR_Billion'].round(2)
    df['MC_INR_Billion'] = df['MC_INR_Billion'].round(2)

    print(df)
    return df
```

6. Output after executing transform function:



```
                                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          124.70          144.96        12929.42
6                   HSBC Holdings PLC          148.90          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
```

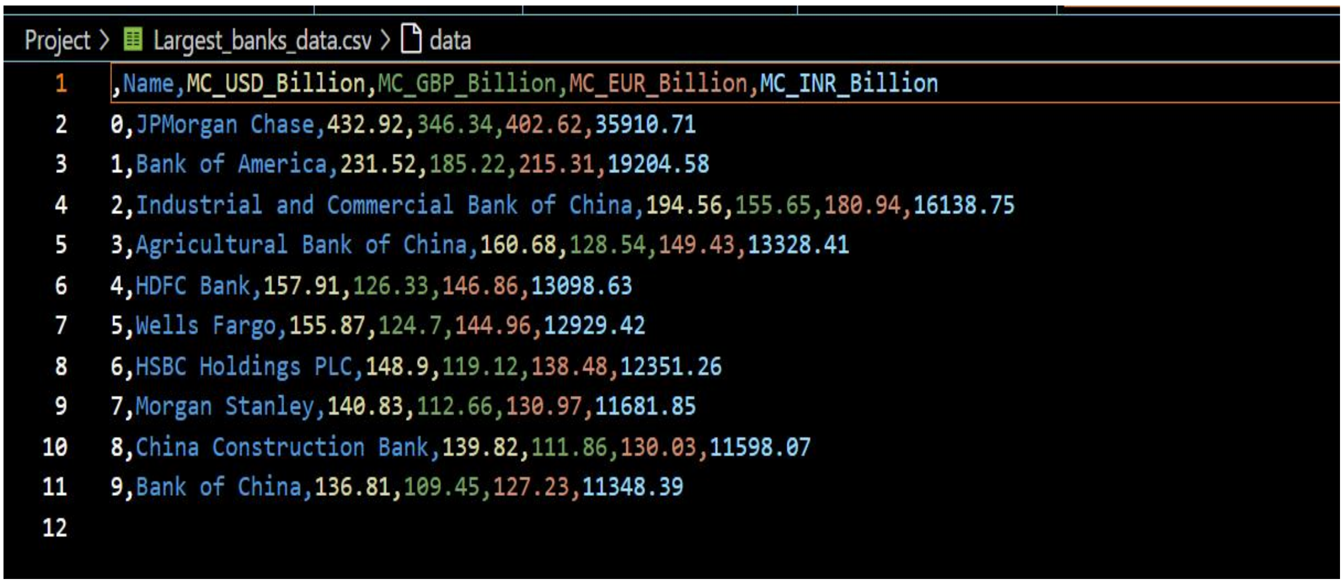
7. Load to csv and db functions code snippet:



Project > banks_project.py > ...

```python
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.'''
    df.to_csv(output_path)


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)

```

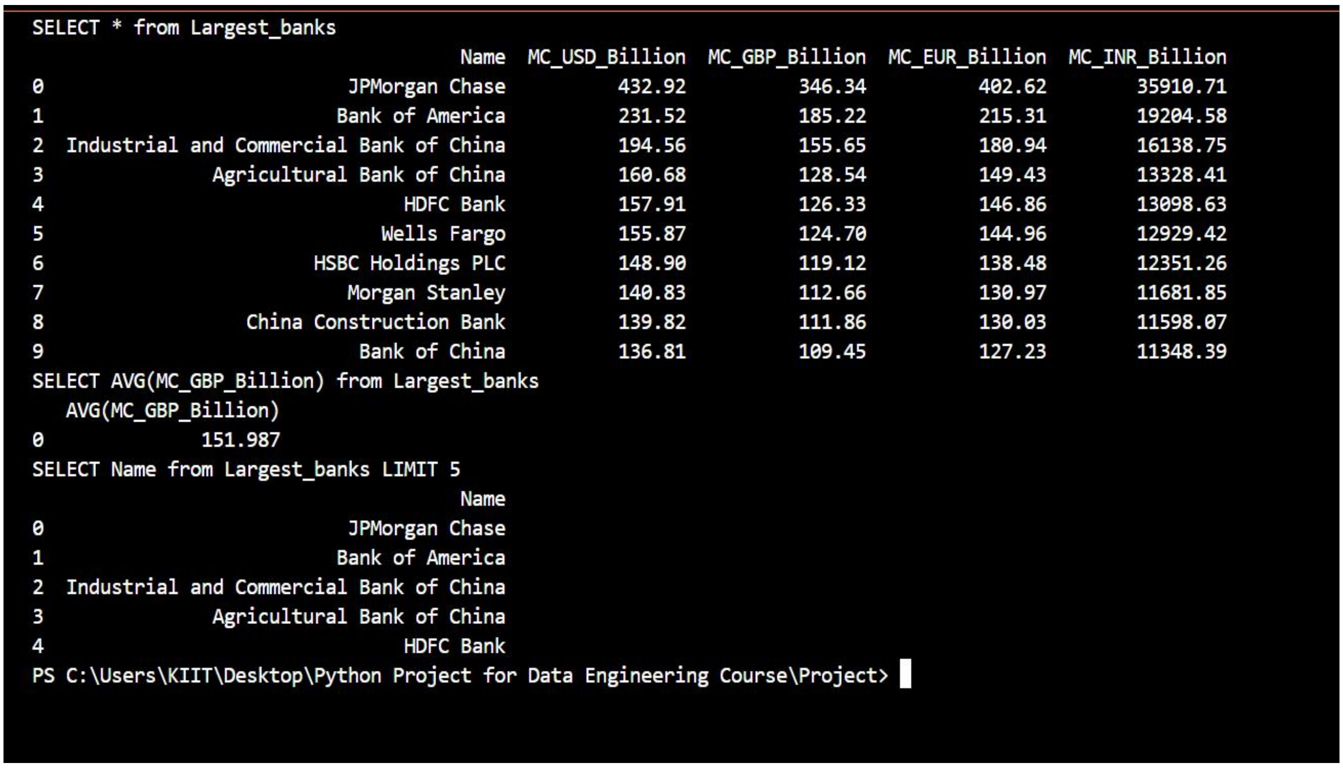
8. CSV file stored after executing the function load_to_csv:



Project > Largest_banks_data.csv > data

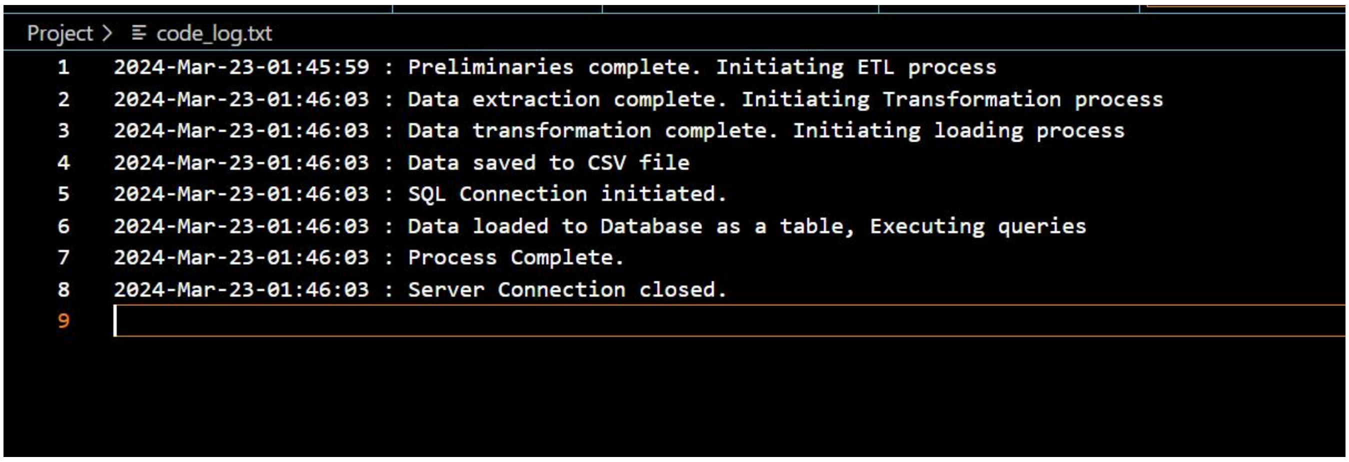
```
,Name,MC_USD_Billion,MC_GBP_Billion,MC_EUR_Billion,MC_INR_Billion
0,JPMorgan Chase,432.92,346.34,402.62,35910.71
1,Bank of America,231.52,185.22,215.31,19204.58
2,Industrial and Commercial Bank of China,194.56,155.65,180.94,16138.75
3,Agricultural Bank of China,160.68,128.54,149.43,13328.41
4,HDFC Bank,157.91,126.33,146.86,13098.63
5,Wells Fargo,155.87,124.7,144.96,12929.42
6,HSBC Holdings PLC,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

```

9. Output after running sql queries using sqlite:



```
SELECT * from Largest_banks
                                     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          124.70          144.96        12929.42
6                       HSBC Holdings PLC          148.90          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
SELECT AVG(MC_GBP_Billion) from Largest_banks
   AVG(MC_GBP_Billion)
0              151.987
SELECT Name from Largest_banks LIMIT 5
                                     Name
0                          JPMorgan Chase
1                         Bank of America
2  Industrial and Commercial Bank of China
3              Agricultural Bank of China
4                               HDFC Bank
PS C:\Users\KIIT\Desktop\Python Project for Data Engineering Course\Project>
```

10. Log File created after running all the steps.



```
2024-Mar-23-01:45:59 : Preliminaries complete. Initiating ETL process
2024-Mar-23-01:46:03 : Data extraction complete. Initiating Transformation process
2024-Mar-23-01:46:03 : Data transformation complete. Initiating loading process
2024-Mar-23-01:46:03 : Data saved to CSV file
2024-Mar-23-01:46:03 : SQL Connection initiated.
2024-Mar-23-01:46:03 : Data loaded to Database as a table, Executing queries
2024-Mar-23-01:46:03 : Process Complete.
2024-Mar-23-01:46:03 : Server Connection closed.
```