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	Name, MC_USD_Billion

Parameter	Value
(upon Extraction only)	
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.

```python

# Importing the required libraries
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import sqlite3
from datetime import datetime

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-%m-%d %H:%M:%S'  # Format for timestamp
    now = datetime.now()  # Get current timestamp
    timestamp = now.strftime(timestamp_format)
    with open("code_log.txt", "a") as f:
        f.write(f'{timestamp} : {message}\n')
```

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

```html
<meta property="mw:PageProp/toc">
<h2> == $0
  <span class="mw-headline" id="By_market_capitalization">By market capitalization</span>
  <span class="mw-editsection">…</span>
</h2>
<p>…</p>
<style data-mw-deduplicate="TemplateStyles:r1096954695/mw-parser-output/.tmulti">…</style>
<div class="thumb tmulti tright">…</div>
<table class="wikitable sortable mw-collapsible jquery-tablesorter mw-made-collapsible">
  <thead>…</thead>
  <tbody>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
    <tr>…</tr>
  </tbody>
  <tfoot></tfoot>
</table>
```

b. Write the code for a function extract() to perform the required data extraction.

theia@theia-spati:/home/project$ python3.11 banks_project.py

Extracted Country: 1, GDP_USD_millions: JPMorgan Chase

Extracted Country: 2, GDP_USD_millions: Bank of America

Extracted Country: 3, GDP_USD_millions: Industrial and Commercial Bank of China

Extracted Country: 4, GDP_USD_millions: Agricultural Bank of China

Extracted Country: 5, GDP_USD_millions: HDFC Bank

Extracted Country: 6, GDP_USD_millions: Wells Fargo

Extracted Country: 7, GDP_USD_millions: HSBC Holdings PLC

Extracted Country: 8, GDP_USD_millions: Morgan Stanley

Extracted Country: 9, GDP_USD_millions: China Construction Bank

Extracted Country: 10, GDP_USD_millions: Bank of China

DataFrame before cleaning:

	Country	GDP_USD_millions
0	1	JPMorgan Chase
1	2	Bank of America
2	3	Industrial and Commercial Bank of China
3	4	Agricultural Bank of China
4	5	HDFC Bank
5	6	Wells Fargo
6	7	HSBC Holdings PLC
7	8	Morgan Stanley
8	9	China Construction Bank
9	10	Bank of China

DataFrame after cleaning:

	Country	GDP_USD_millions
0	1	
1	2	
2	3	
3	4	
4	5	

```
5     6
6     7
7     8
8     9
9    10
Final DataFrame:
  Country  GDP_USD_millions
0     1           NaN
1     2           NaN
2     3           NaN
3     4           NaN
4     5           NaN
5     6           NaN
6     7           NaN
7     8           NaN
8     9           NaN
9    10            NaN
  Country  GDP_USD_millions
0     1           NaN
1     2           NaN
2     3           NaN
3     4           NaN
4     5           NaN
5     6           NaN
6     7           NaN
7     8           NaN
8     9           NaN
9    10            NaN
theia@theia-spati:/home/project$ python3.11 banks_project.py
```

Extracted Country: 1, GDP_USD_millions: JPMorgan Chase

Extracted Country: 2, GDP_USD_millions: Bank of America

Extracted Country: 3, GDP_USD_millions: Industrial and Commercial Bank of China

Extracted Country: 4, GDP_USD_millions: Agricultural Bank of China

Extracted Country: 5, GDP_USD_millions: HDFC Bank

Extracted Country: 6, GDP_USD_millions: Wells Fargo

Extracted Country: 7, GDP_USD_millions: HSBC Holdings PLC

Extracted Country: 8, GDP_USD_millions: Morgan Stanley

Extracted Country: 9, GDP_USD_millions: China Construction Bank

Extracted Country: 10, GDP_USD_millions: Bank of China

DataFrame before cleaning:

	Country	GDP_USD_millions
0	1	JPMorgan Chase
1	2	Bank of America
2	3	Industrial and Commercial Bank of China
3	4	Agricultural Bank of China
4	5	HDFC Bank
5	6	Wells Fargo
6	7	HSBC Holdings PLC
7	8	Morgan Stanley
8	9	China Construction Bank
9	10	Bank of China

Column names: Index(['Country', 'GDP_USD_millions'], dtype='object')

DataFrame after cleaning:

	Country	GDP_USD_millions
0	1	
1	2	
2	3	
3	4	

```
4     5
5     6
6     7
7     8
8     9
9    10
Final DataFrame:
  Country  GDP_USD_millions
0     1           NaN
1     2           NaN
2     3           NaN
3     4           NaN
4     5           NaN
5     6           NaN
6     7           NaN
7     8           NaN
8     9           NaN
9    10            NaN
  Country  GDP_USD_millions
0     1           NaN
1     2           NaN
2     3           NaN
3     4           NaN
4     5           NaN
5     6           NaN
6     7           NaN
7     8           NaN
8     9           NaN
9    10            NaN
```

```
theia@theia-spati:/home/project$ python3.11 banks_project.py
Extracted Rank: 1, Bank Name: JPMorgan Chase, Market Cap: 432.92
Extracted Rank: 2, Bank Name: Bank of America, Market Cap: 231.52
Extracted Rank: 3, Bank Name: Industrial and Commercial Bank of China, Market Cap: 194.56
Extracted Rank: 4, Bank Name: Agricultural Bank of China, Market Cap: 160.68
Extracted Rank: 5, Bank Name: HDFC Bank, Market Cap: 157.91
Extracted Rank: 6, Bank Name: Wells Fargo, Market Cap: 155.87
Extracted Rank: 7, Bank Name: HSBC Holdings PLC, Market Cap: 148.90
Extracted Rank: 8, Bank Name: Morgan Stanley, Market Cap: 140.83
Extracted Rank: 9, Bank Name: China Construction Bank, Market Cap: 139.82
Extracted Rank: 10, Bank Name: Bank of China, Market Cap: 136.81
DataFrame before cleaning:
  Rank                        Bank_Name Market_Cap
0    1                     JPMorgan Chase     432.92
1    2                    Bank of America     231.52
2    3  Industrial and Commercial Bank of China     194.56
3    4            Agricultural Bank of China     160.68
4    5                          HDFC Bank     157.91
5    6                        Wells Fargo     155.87
6    7                   HSBC Holdings PLC     148.90
7    8                     Morgan Stanley     140.83
8    9              China Construction Bank     139.82
9   10                       Bank of China     136.81
Column names: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
DataFrame after cleaning:
  Rank                        Bank_Name Market_Cap
0    1                     JPMorgan Chase     432.92
1    2                    Bank of America     231.52
2    3  Industrial and Commercial Bank of China     194.56
```

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Final DataFrame:

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

```
9  10                Bank of China     136.81
theia@theia-spati:/home/project$ python3.11 banks_project.py
Extracted Rank: 1, Bank Name: JPMorgan Chase, Market Cap: 432.92
Extracted Rank: 2, Bank Name: Bank of America, Market Cap: 231.52
Extracted Rank: 3, Bank Name: Industrial and Commercial Bank of China, Market Cap: 194.56
Extracted Rank: 4, Bank Name: Agricultural Bank of China, Market Cap: 160.68
Extracted Rank: 5, Bank Name: HDFC Bank, Market Cap: 157.91
Extracted Rank: 6, Bank Name: Wells Fargo, Market Cap: 155.87
Extracted Rank: 7, Bank Name: HSBC Holdings PLC, Market Cap: 148.90
Extracted Rank: 8, Bank Name: Morgan Stanley, Market Cap: 140.83
Extracted Rank: 9, Bank Name: China Construction Bank, Market Cap: 139.82
Extracted Rank: 10, Bank Name: Bank of China, Market Cap: 136.81
DataFrame before cleaning:
  Rank                Bank_Name Market_Cap
0    1             JPMorgan Chase    432.92
1    2             Bank of America    231.52
2    3  Industrial and Commercial Bank of China    194.56
3    4        Agricultural Bank of China    160.68
4    5                 HDFC Bank    157.91
5    6              Wells Fargo    155.87
6    7            HSBC Holdings PLC    148.90
7    8             Morgan Stanley    140.83
8    9         China Construction Bank    139.82
9   10              Bank of China    136.81
Column names: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
DataFrame after cleaning:
  Rank                Bank_Name Market_Cap
0    1             JPMorgan Chase    432.92
1    2             Bank of America    231.52
```

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Final DataFrame:

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

theia@theia-spati:/home/project$

c. Execute a function call to extract() to verify the output.

```
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.'''
    # Request and parse the webpage
    page = requests.get(url).text
    soup = BeautifulSoup(page, 'html.parser')

    # Locate the table under the heading "By market capitalization"
    heading = soup.find('span', {'id': 'By_market_capitalization'})
    if heading:
        table = heading.find_next('table', {'class': 'wikitable'})
    else:
        print("Table heading not found")
        return pd.DataFrame(columns=table_attribs)

    # Initialize an empty DataFrame with the expected columns
    df = pd.DataFrame(columns=table_attribs)

    # Extract data from the table rows
    rows = table.find_all('tr')
    data_list = []
    for row in rows[1:]:  # Skip the header row
        cols = row.find_all('td')
        if len(cols) >= 3:  # Ensure there are enough columns
            # Extract columns: Rank, Bank Name, Market Cap
            rank = cols[0].get_text(strip=True)
            bank_name = cols[1].get_text(strip=True)
            market_cap = cols[2].get_text(strip=True).replace(' US$ billion', '').strip()

            # Print extracted data for debugging
            print(f"Extracted Rank: {rank}, Bank Name: {bank_name}, Market Cap: {market_cap}")

            # Create a dictionary for the row
            data_dict = {"Rank": rank, "Bank_Name": bank_name, "Market_Cap": market_cap}
            data_list.append(data_dict)

    # Create DataFrame from the collected data
    df = pd.DataFrame(data_list, columns=table_attribs)

    # Print DataFrame before cleaning for debugging
    print("DataFrame before cleaning:")
    print(df)

    # Check column names and structure
    print("Column names:", df.columns)

    # Clean and convert the 'Market_Cap' column
    df['Market_Cap'] = df['Market_Cap'].str.replace('[^\d.]', '', regex=True)

    # Print DataFrame after cleaning for debugging
    print("DataFrame after cleaning:")
    print(df)

    # Convert to float and handle errors
    def safe_float(x):
        try:
            return float(x)
        except ValueError:
```

theia@theia-spati: /home/project

```
DataFrame after cleaning:
  Rank                                Bank_Name Market_Cap
0    1                            JPMorgan Chase     432.92
1    2                           Bank of America     231.52
2    3  Industrial and Commercial Bank of China     194.56
3    4                Agricultural Bank of China     160.68
4    5                                  HDFC Bank     157.91
5    6                               Wells Fargo     155.87
6    7                        HSBC Holdings PLC     148.90
7    8                           Morgan Stanley     140.83
8    9                   China Construction Bank     139.82
9   10                            Bank of China     136.81
Final DataFrame:
  Rank                                Bank_Name  Market_Cap
0    1                            JPMorgan Chase      432.92
1    2                           Bank of America      231.52
2    3  Industrial and Commercial Bank of China      194.56
3    4                Agricultural Bank of China      160.68
4    5                                  HDFC Bank      157.91
5    6                               Wells Fargo      155.87
6    7                        HSBC Holdings PLC      148.90
7    8                           Morgan Stanley      140.83
8    9                   China Construction Bank      139.82
9   10                            Bank of China      136.81
  Rank                                Bank_Name  Market_Cap
0    1                            JPMorgan Chase      432.92
1    2                           Bank of America      231.52
2    3  Industrial and Commercial Bank of China      194.56
3    4                Agricultural Bank of China      160.68
4    5                                  HDFC Bank      157.91
5    6                               Wells Fargo      155.87
6    7                        HSBC Holdings PLC      148.90
7    8                           Morgan Stanley      140.83
8    9                   China Construction Bank      139.82
9   10                            Bank of China      136.81
theia@theia-spati:/home/project$
```

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.

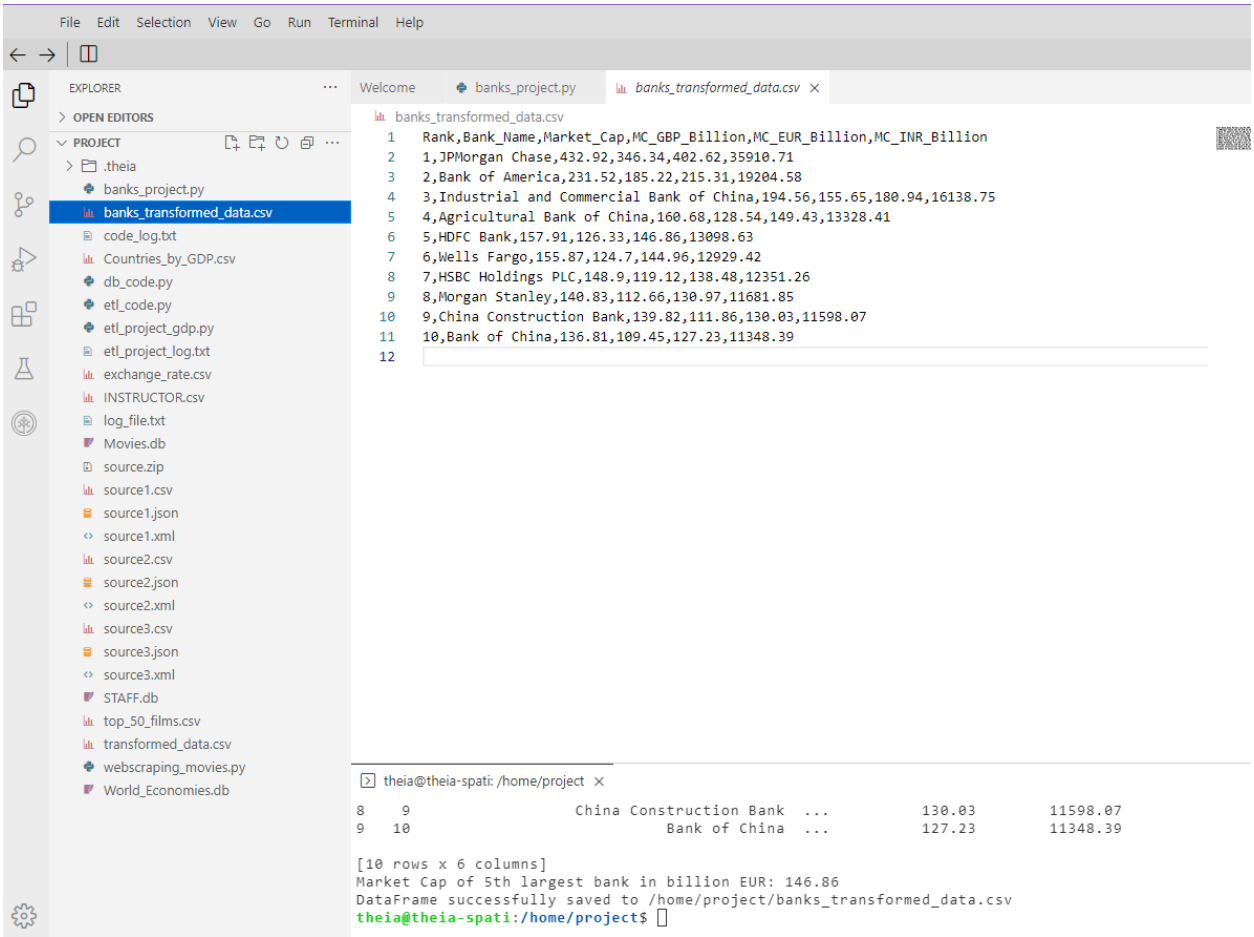
```python
def transform(df, exchange_rate_csv):
    '''This function transforms the DataFrame by adding columns for market capitalization
    in different currencies based on the exchange rates from the CSV file.'''

    # Read the exchange rate CSV file and convert it to a dictionary
    exchange_rate_df = pd.read_csv(exchange_rate_csv)
    exchange_rate = exchange_rate_df.set_index('Currency').to_dict()['Rate']

    # Print DataFrame columns for debugging
    print("DataFrame columns:", df.columns)

    # Ensure the column name is correct
    if 'Market_Cap' not in df.columns:
        raise KeyError("The DataFrame does not contain the 'Market_Cap' column")

    # Add columns for market capitalization in GBP, EUR, and INR
    df['MC_GBP_Billion'] = [np.round(x * exchange_rate.get('GBP', 0), 2) for x in df['Market_Cap']]
    df['MC_EUR_Billion'] = [np.round(x * exchange_rate.get('EUR', 0), 2) for x in df['Market_Cap']]
    df['MC_INR_Billion'] = [np.round(x * exchange_rate.get('INR', 0), 2) for x in df['Market_Cap']]

    return df
```

b. Execute a function call to transform() and verify the output.

```
Final DataFrame:
  Rank                              Bank_Name  Market_Cap
0    1                           JPMorgan Chase      432.92
1    2                          Bank of America      231.52
2    3  Industrial and Commercial Bank of China      194.56
3    4               Agricultural Bank of China      160.68
4    5                                 HDFC Bank      157.91
5    6                              Wells Fargo      155.87
6    7                         HSBC Holdings PLC      148.90
7    8                            Morgan Stanley      140.83
8    9                   China Construction Bank      139.82
9   10                            Bank of China      136.81
  Rank                              Bank_Name  Market_Cap
0    1                           JPMorgan Chase      432.92
1    2                          Bank of America      231.52
2    3  Industrial and Commercial Bank of China      194.56
3    4               Agricultural Bank of China      160.68
4    5                                 HDFC Bank      157.91
5    6                              Wells Fargo      155.87
6    7                         HSBC Holdings PLC      148.90
7    8                            Morgan Stanley      140.83
8    9                   China Construction Bank      139.82
9   10                            Bank of China      136.81
DataFrame columns: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
  Rank                              Bank_Name  Market_Cap  MC_GBP_Billion  MC_EUR_Billion  MC_INR_Billion
0    1                           JPMorgan Chase      432.92          346.34          402.62        35910.71
1    2                          Bank of America      231.52          185.22          215.31        19204.58
2    3  Industrial and Commercial Bank of China      194.56          155.65          180.94        16138.75
3    4               Agricultural Bank of China      160.68          128.54          149.43        13328.41
4    5                                 HDFC Bank      157.91          126.33          146.86        13098.63
5    6                              Wells Fargo      155.87          124.70          144.96        12929.42
6    7                         HSBC Holdings PLC      148.90          119.12          138.48        12351.26
7    8                            Morgan Stanley      140.83          112.66          130.97        11681.85
8    9                   China Construction Bank      139.82          111.86          130.03        11598.07
9   10                            Bank of China      136.81          109.45          127.23        11348.39
Market Cap of 5th largest bank in billion EUR: 146.86
```

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.

EXPLORER

OPEN EDITORS

PROJECT

- .theia
- banks_project.py
- banks_transformed_data.csv
- Banks.db
- code_log.txt
- Countries_by_GDP.csv
- db_code.py
- etl_code.py
- etl_project_gdp.py
- etl_project_log.txt
- exchange_rate.csv
- INSTRUCTOR.csv
- log_file.txt
- Movies.db
- source.zip
- source1.csv
- source1.json
- source1.xml
- source2.csv
- source2.json
- source2.xml

Welcome | banks_project.py | banks_transformed_data.csv

banks_project.py

```python
    df['MC_INR_Billion'] = [np.round(x * exchange_rate.get('INR', 0), 2) for x in df['Market_C

    return df



def load_to_csv(df, file_path):
    """
    Save the DataFrame to a CSV file.

    :param df: DataFrame to save
    :param file_path: Path to save the CSV file
    """
    df.to_csv(file_path, index=False)
    print(f"DataFrame successfully saved to {file_path}")


def load_to_db(df, sql_connection, table_name):
    try:
        df.to_sql(table_name, sql_connection, if_exists='replace', index=False)
        print(f"DataFrame successfully loaded into the table '{table_name}' in the database.")
        log_progress(f"Data successfully loaded into '{table_name}' table.")
    except Exception as e:
        print(f"An error occurred while loading data to the database: {e}")
        log_progress(f"Error loading data to the '{table_name}' table: {e}")
```

Task 6:

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

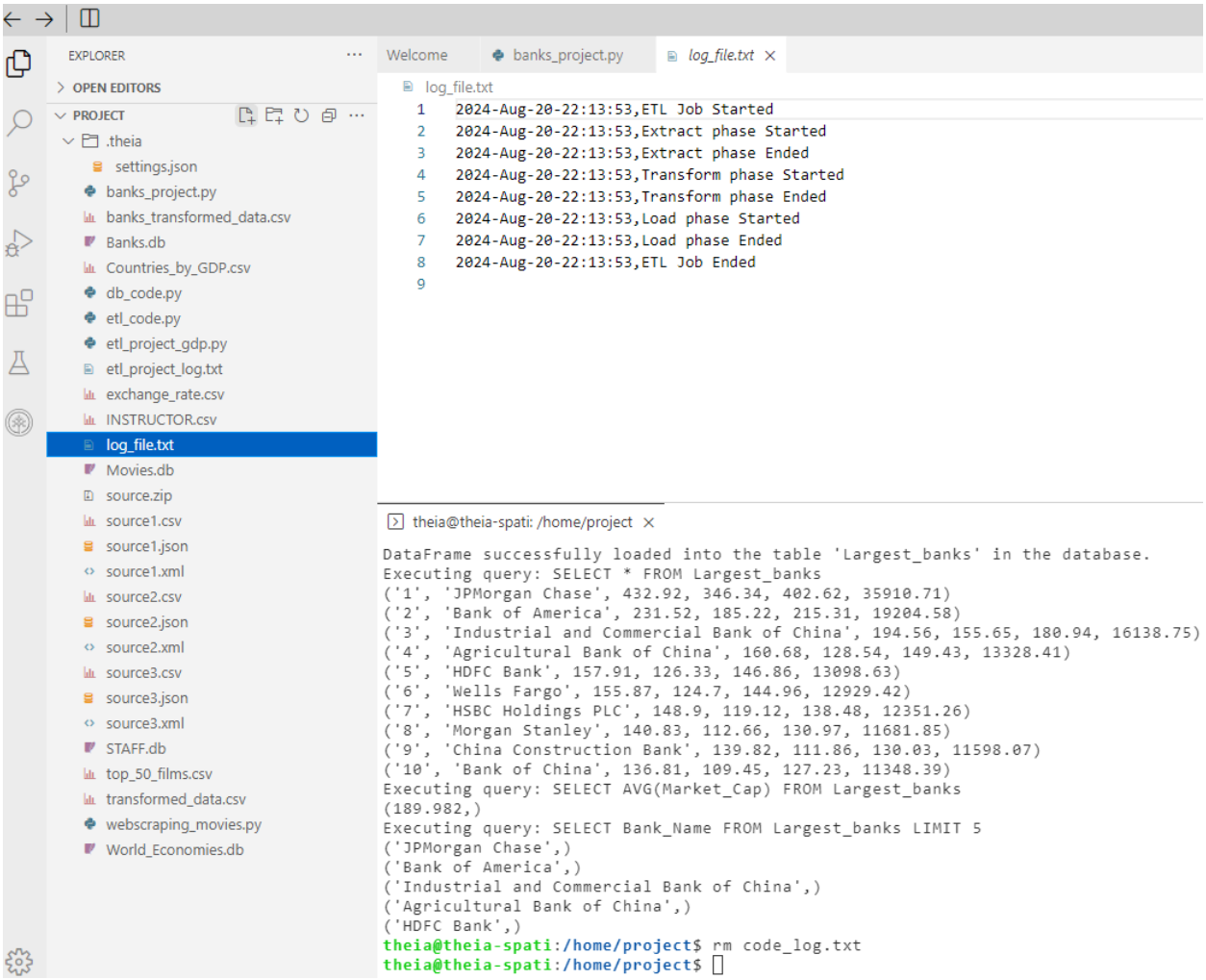
theia@theia-spati: /home/project

Terminal 1

```
D[illegible]uccessfully saved to /home/project/banks_transformed_data.csv
DataFrame successfully loaded into the table 'Largest_banks' in the database.
Executing query: SELECT * FROM Largest_banks
('1', 'JPMorgan Chase', 432.92, 346.34, 402.62, 35910.71)
('2', 'Bank of America', 231.52, 185.22, 215.31, 19204.58)
('3', 'Industrial and Commercial Bank of China', 194.56, 155.65, 180.94, 16138.75)
('4', 'Agricultural Bank of China', 160.68, 128.54, 149.43, 13328.41)
('5', 'HDFC Bank', 157.91, 126.33, 146.86, 13098.63)
('6', 'Wells Fargo', 155.87, 124.7, 144.96, 12929.42)
('7', 'HSBC Holdings PLC', 148.9, 119.12, 138.48, 12351.26)
('8', 'Morgan Stanley', 140.83, 112.66, 130.97, 11681.85)
('9', 'China Construction Bank', 139.82, 111.86, 130.03, 11598.07)
('10', 'Bank of China', 136.81, 109.45, 127.23, 11348.39)
Executing query: SELECT AVG(Market_Cap) FROM Largest_banks
(189.982,)
Executing query: SELECT Bank_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-spati:/home/project$
```

Task 7:

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



Code :

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


# Importing the required libraries

import requests

from bs4 import BeautifulSoup

import pandas as pd

import numpy as np

import sqlite3

from datetime import datetime
```

```
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-%m-%d %H:%M:%S'  # Format for timestamp
    now = datetime.now()  # Get current timestamp
    timestamp = now.strftime(timestamp_format)
    with open("code_log.txt", "a") as f:
        f.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.'''
    # Request and parse the webpage
    page = requests.get(url).text
    soup = BeautifulSoup(page, 'html.parser')

    # Locate the table under the heading "By market capitalization"
    heading = soup.find('span', {'id': 'By_market_capitalization'})
    if heading:
        table = heading.find_next('table', {'class': 'wikitable'})
    else:
        print("Table heading not found")
        return pd.DataFrame(columns=table_attribs)

    # Initialize an empty DataFrame with the expected columns
    df = pd.DataFrame(columns=table_attribs)

    # Extract data from the table rows
```

```
rows = table.find_all('tr')
data_list = []
for row in rows[1:]:  # Skip the header row
    cols = row.find_all('td')
    if len(cols) >= 3:  # Ensure there are enough columns
        # Extract columns: Rank, Bank Name, Market Cap
        rank = cols[0].get_text(strip=True)
        bank_name = cols[1].get_text(strip=True)
        market_cap = cols[2].get_text(strip=True).replace(' US$ billion', '').strip()

        # Print extracted data for debugging
        print(f"Extracted Rank: {rank}, Bank Name: {bank_name}, Market Cap: {market_cap}")

        # Create a dictionary for the row
        data_dict = {"Rank": rank, "Bank_Name": bank_name, "Market_Cap": market_cap}
        data_list.append(data_dict)

# Create DataFrame from the collected data
df = pd.DataFrame(data_list, columns=table_attribs)

# Print DataFrame before cleaning for debugging
print("DataFrame before cleaning:")
print(df)

# Check column names and structure
print("Column names:", df.columns)

# Clean and convert the 'Market_Cap' column
df['Market_Cap'] = df['Market_Cap'].str.replace('[^\d.]', '', regex=True)
```

```
    # Print DataFrame after cleaning for debugging
    print("DataFrame after cleaning:")
    print(df)

    # Convert to float and handle errors
    def safe_float(x):
        try:
            return float(x)
        except ValueError:
            return np.nan  # Return NaN for invalid values

    df['Market_Cap'] = df['Market_Cap'].apply(safe_float)

    # Print final DataFrame for debugging
    print("Final DataFrame:")
    print(df)

    return df

def transform(df, exchange_rate_csv):
    '''This function transforms the DataFrame by adding columns for market capitalization
    in different currencies based on the exchange rates from the CSV file.'''

    # Read the exchange rate CSV file and convert it to a dictionary
    exchange_rate_df = pd.read_csv(exchange_rate_csv)
    exchange_rate = exchange_rate_df.set_index('Currency').to_dict()['Rate']

    # Print DataFrame columns for debugging
```

```
    print("DataFrame columns:", df.columns)

    # Ensure the column name is correct
    if 'Market_Cap' not in df.columns:
        raise KeyError("The DataFrame does not contain the 'Market_Cap' column")

    # Add columns for market capitalization in GBP, EUR, and INR
    df['MC_GBP_Billion'] = [np.round(x * exchange_rate.get('GBP', 0), 2) for x in df['Market_Cap']]
    df['MC_EUR_Billion'] = [np.round(x * exchange_rate.get('EUR', 0), 2) for x in df['Market_Cap']]
    df['MC_INR_Billion'] = [np.round(x * exchange_rate.get('INR', 0), 2) for x in df['Market_Cap']]

    return df


def load_to_csv(df, file_path):
    """
    Save the DataFrame to a CSV file.

    :param df: DataFrame to save
    :param file_path: Path to save the CSV file
    """
    df.to_csv(file_path, index=False)
    print(f"DataFrame successfully saved to {file_path}")

def load_to_db(df, sql_connection, table_name):
    try:
        df.to_sql(table_name, sql_connection, if_exists='replace', index=False)
        print(f"DataFrame successfully loaded into the table '{table_name}' in the database.")
        log_progress(f"Data successfully loaded into '{table_name}' table.")
```

```
    except Exception as e:
        print(f"An error occurred while loading data to the database: {e}")
        log_progress(f"Error loading data to the '{table_name}' table: {e}")

def run_queries(query_statement, sql_connection):
    '''This function runs the query on the database table and
    prints the output on the terminal.'''
    try:
        # Print the query statement
        print(f"Executing query: {query_statement}")
        log_progress(f"Executing query: {query_statement}")

        # Execute the query
        cursor = sql_connection.cursor()
        cursor.execute(query_statement)

        # Fetch all results
        results = cursor.fetchall()

        # Print the results
        for row in results:
            print(row)

        log_progress("Query executed successfully.")
    except Exception as e:
        print(f"An error occurred while executing the query: {e}")
        log_progress(f"Error executing query: {e}")

# URL and table attributes
```

```
url =
'https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List_of_largest_banks'

table_attribs = ["Rank", "Bank_Name", "Market_Cap"]


# Extract the data

df = extract(url, table_attribs)


# Print the DataFrame to verify

print(df)


# Log the progress

log_progress('Data extraction complete. Initiating Transformation process')


# Transform the data

exchange_rate_csv = 'file:///home/project/exchange_rate.csv'

df_transformed = transform(df, exchange_rate_csv)


# Print the DataFrame to verify the results

print(df_transformed)


# Print the 5th largest bank's market cap in EUR

print("Market Cap of 5th largest bank in billion EUR:", df_transformed['MC_EUR_Billion'][4])


# Log the progress

log_progress('Data transformation complete. Saving to CSV')


# Save the transformed DataFrame to CSV

csv_file_path = '/home/project/banks_transformed_data.csv'

load_to_csv(df_transformed, csv_file_path)
```

```
# Connect to the SQLite3 database
connection = sqlite3.connect('/home/project/Banks.db')

# Log the progress
log_progress('Database connection established.')

# Define the table name
table_name = 'Largest_banks'

# Load the data into the database
load_to_db(df_transformed, connection, table_name)

# Log the progress
log_progress('Data loading to the database complete.')

# Connect to the SQLite3 database
connection = sqlite3.connect('/home/project/Banks.db')

# Query 1: Print the contents of the entire table
query_1 = "SELECT * FROM Largest_banks"
run_queries(query_1, connection)

# Query 2: Print the average market capitalization of all the banks in Billion USD
query_2 = "SELECT AVG(Market_Cap) FROM Largest_banks"
run_queries(query_2, connection)

# Query 3: Print only the names of the top 5 banks
query_3 = "SELECT Bank_Name FROM Largest_banks LIMIT 5"
```

run_queries(query_3, connection)

whole output:

```
theia@theia-spati:/home/project$ python3.11 banks_project.py
Extracted Rank: 1, Bank Name: JPMorgan Chase, Market Cap: 432.92
Extracted Rank: 2, Bank Name: Bank of America, Market Cap: 231.52
Extracted Rank: 3, Bank Name: Industrial and Commercial Bank of China, Market Cap: 194.56
Extracted Rank: 4, Bank Name: Agricultural Bank of China, Market Cap: 160.68
Extracted Rank: 5, Bank Name: HDFC Bank, Market Cap: 157.91
Extracted Rank: 6, Bank Name: Wells Fargo, Market Cap: 155.87
Extracted Rank: 7, Bank Name: HSBC Holdings PLC, Market Cap: 148.90
Extracted Rank: 8, Bank Name: Morgan Stanley, Market Cap: 140.83
Extracted Rank: 9, Bank Name: China Construction Bank, Market Cap: 139.82
Extracted Rank: 10, Bank Name: Bank of China, Market Cap: 136.81
DataFrame before cleaning:
  Rank                          Bank_Name Market_Cap
0    1                       JPMorgan Chase     432.92
1    2                      Bank of America     231.52
2    3  Industrial and Commercial Bank of China     194.56
3    4            Agricultural Bank of China     160.68
4    5                          HDFC Bank     157.91
5    6                         Wells Fargo     155.87
6    7                    HSBC Holdings PLC     148.90
7    8                       Morgan Stanley     140.83
8    9               China Construction Bank     139.82
9   10                        Bank of China     136.81
Column names: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
DataFrame after cleaning:
  Rank                          Bank_Name Market_Cap
```

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Final DataFrame:

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

6 7 HSBC Holdings PLC 148.90

7 8 Morgan Stanley 140.83

8 9 China Construction Bank 139.82

9 10 Bank of China 136.81

Rank Bank_Name Market_Cap

0 1 JPMorgan Chase 432.92

1 2 Bank of America 231.52

2 3 Industrial and Commercial Bank of China 194.56

3 4 Agricultural Bank of China 160.68

4 5 HDFC Bank 157.91

5 6 Wells Fargo 155.87

```
6   7                HSBC Holdings PLC     148.90
7   8                  Morgan Stanley     140.83
8   9           China Construction Bank     139.82
9  10                   Bank of China     136.81
DataFrame columns: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
  Rank                      Bank_Name  ...  MC_EUR_Billion  MC_INR_Billion
0   1                   JPMorgan Chase  ...        402.62       35910.71
1   2                  Bank of America  ...        215.31       19204.58
2   3  Industrial and Commercial Bank of China  ...        180.94       16138.75
3   4          Agricultural Bank of China  ...        149.43       13328.41
4   5                      HDFC Bank  ...        146.86       13098.63
5   6                    Wells Fargo  ...        144.96       12929.42
6   7                HSBC Holdings PLC  ...        138.48       12351.26
7   8                  Morgan Stanley  ...        130.97       11681.85
8   9           China Construction Bank  ...        130.03       11598.07
9  10                   Bank of China  ...        127.23       11348.39

[10 rows x 6 columns]
Market Cap of 5th largest bank in billion EUR: 146.86
DataFrame successfully saved to /home/project/banks_transformed_data.csv
DataFrame successfully loaded into the table 'Largest_banks' in the database.
theia@theia-spati:/home/project$ SELECT * FROM Largest_banks
bash: SELECT: command not found
theia@theia-spati:/home/project$ python3.11 banks_project,py
python3.11: can't open file '/home/project/banks_project,py': [Errno 2] No such file or directory
theia@theia-spati:/home/project$ python3.11 banks_project.py
Extracted Rank: 1, Bank Name: JPMorgan Chase, Market Cap: 432.92
Extracted Rank: 2, Bank Name: Bank of America, Market Cap: 231.52
Extracted Rank: 3, Bank Name: Industrial and Commercial Bank of China, Market Cap: 194.56
```

Extracted Rank: 4, Bank Name: Agricultural Bank of China, Market Cap: 160.68

Extracted Rank: 5, Bank Name: HDFC Bank, Market Cap: 157.91

Extracted Rank: 6, Bank Name: Wells Fargo, Market Cap: 155.87

Extracted Rank: 7, Bank Name: HSBC Holdings PLC, Market Cap: 148.90

Extracted Rank: 8, Bank Name: Morgan Stanley, Market Cap: 140.83

Extracted Rank: 9, Bank Name: China Construction Bank, Market Cap: 139.82

Extracted Rank: 10, Bank Name: Bank of China, Market Cap: 136.81

DataFrame before cleaning:

	Rank	Bank_Name	Market_Cap
0	1	JPMorgan Chase	432.92
1	2	Bank of America	231.52
2	3	Industrial and Commercial Bank of China	194.56
3	4	Agricultural Bank of China	160.68
4	5	HDFC Bank	157.91
5	6	Wells Fargo	155.87
6	7	HSBC Holdings PLC	148.90
7	8	Morgan Stanley	140.83
8	9	China Construction Bank	139.82
9	10	Bank of China	136.81

Column names: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')

DataFrame after cleaning:

	Rank	Bank_Name	Market_Cap
0	1	JPMorgan Chase	432.92
1	2	Bank of America	231.52
2	3	Industrial and Commercial Bank of China	194.56
3	4	Agricultural Bank of China	160.68
4	5	HDFC Bank	157.91
5	6	Wells Fargo	155.87
6	7	HSBC Holdings PLC	148.90

```
7   8                Morgan Stanley    140.83
8   9          China Construction Bank    139.82
9  10                Bank of China    136.81
Final DataFrame:
  Rank                    Bank_Name  Market_Cap
0   1                JPMorgan Chase     432.92
1   2                Bank of America     231.52
2   3  Industrial and Commercial Bank of China     194.56
3   4         Agricultural Bank of China     160.68
4   5                  HDFC Bank     157.91
5   6                Wells Fargo     155.87
6   7              HSBC Holdings PLC     148.90
7   8                Morgan Stanley     140.83
8   9          China Construction Bank     139.82
9  10                Bank of China     136.81
  Rank                    Bank_Name  Market_Cap
0   1                JPMorgan Chase     432.92
1   2                Bank of America     231.52
2   3  Industrial and Commercial Bank of China     194.56
3   4         Agricultural Bank of China     160.68
4   5                  HDFC Bank     157.91
5   6                Wells Fargo     155.87
6   7              HSBC Holdings PLC     148.90
7   8                Morgan Stanley     140.83
8   9          China Construction Bank     139.82
9  10                Bank of China     136.81
DataFrame columns: Index(['Rank', 'Bank_Name', 'Market_Cap'], dtype='object')
  Rank                    Bank_Name  ...  MC_EUR_Billion  MC_INR_Billion
0   1                JPMorgan Chase  ...        402.62       35910.71
```

```
1   2                    Bank of America  ...        215.31       19204.58
2   3  Industrial and Commercial Bank of China  ...        180.94       16138.75
3   4           Agricultural Bank of China  ...        149.43       13328.41
4   5                      HDFC Bank  ...        146.86       13098.63
5   6                     Wells Fargo  ...        144.96       12929.42
6   7                 HSBC Holdings PLC  ...        138.48       12351.26
7   8                   Morgan Stanley  ...        130.97       11681.85
8   9             China Construction Bank  ...        130.03       11598.07
9  10                    Bank of China  ...        127.23       11348.39

[10 rows x 6 columns]
Market Cap of 5th largest bank in billion EUR: 146.86
DataFrame successfully saved to /home/project/banks_transformed_data.csv
DataFrame successfully loaded into the table 'Largest_banks' in the database.
Executing query: SELECT * FROM Largest_banks
('1', 'JPMorgan Chase', 432.92, 346.34, 402.62, 35910.71)
('2', 'Bank of America', 231.52, 185.22, 215.31, 19204.58)
('3', 'Industrial and Commercial Bank of China', 194.56, 155.65, 180.94, 16138.75)
('4', 'Agricultural Bank of China', 160.68, 128.54, 149.43, 13328.41)
('5', 'HDFC Bank', 157.91, 126.33, 146.86, 13098.63)
('6', 'Wells Fargo', 155.87, 124.7, 144.96, 12929.42)
('7', 'HSBC Holdings PLC', 148.9, 119.12, 138.48, 12351.26)
('8', 'Morgan Stanley', 140.83, 112.66, 130.97, 11681.85)
('9', 'China Construction Bank', 139.82, 111.86, 130.03, 11598.07)
('10', 'Bank of China', 136.81, 109.45, 127.23, 11348.39)
Executing query: SELECT AVG(Market_Cap) FROM Largest_banks
(189.982,)
Executing query: SELECT Bank_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-spati:/home/project$ rm code_log.txt

theia@theia-spati:/home/project$

log file:

2024-Aug-20-22:13:53,ETL Job Started

2024-Aug-20-22:13:53,Extract phase Started

2024-Aug-20-22:13:53,Extract phase Ended

2024-Aug-20-22:13:53,Transform phase Started

2024-Aug-20-22:13:53,Transform phase Ended

2024-Aug-20-22:13:53,Load phase Started

2024-Aug-20-22:13:53,Load phase Ended

2024-Aug-20-22:13:53,ETL Job Ended

Bank transformed data:

Rank,Bank_Name,Market_Cap,MC_GBP_Billion,MC_EUR_Billion,MC_INR_Billion

1,JPMorgan Chase,432.92,346.34,402.62,35910.71

2,Bank of America,231.52,185.22,215.31,19204.58

3,Industrial and Commercial Bank of China,194.56,155.65,180.94,16138.75

4,Agricultural Bank of China,160.68,128.54,149.43,13328.41

5,HDFC Bank,157.91,126.33,146.86,13098.63

6,Wells Fargo,155.87,124.7,144.96,12929.42

7,HSBC Holdings PLC,148.9,119.12,138.48,12351.26

8,Morgan Stanley,140.83,112.66,130.97,11681.85

9,China Construction Bank,139.82,111.86,130.03,11598.07

10,Bank of China,136.81,109.45,127.23,11348.39

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