**Extraction of Data - Adım Adım**

**Hedef**

* URL’den (https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List\_of\_largest\_banks) "By market capitalization" başlığı altındaki tabloyu analiz et.
* Tabloyu bir Pandas DataFrame’e çek.
* Market Cap sütunundaki \n karakterini kaldırıp değeri float’a çevir.
* Fonksiyonu çağırıp çıktıyı yazdır.
* Log kaydı yap.

**Adım 1: Web Sayfasını Analiz Et**

* **Talimat**: Web sayfasını incele, "By market capitalization" altındaki tablonun yerini ve HTML yapısını belirle.
* **Yapılacaklar**:
  1. Tarayıcıda URL’yi aç (https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List\_of\_largest\_banks).
  2. "By market capitalization" başlığını bul (genelde <span> veya <h2> etiketiyle işaretlenmiş).
  3. İlgili tabloyu incele (genelde <table> etiketi içinde, <tr> ve <td> ile satır/sütunlar).
  4. **Gözlem**: Tablonun <table> etiketleri arasında, başlıktan sonra geliyor. İlk sütun sıralama, ikinci sütun banka adı, üçüncü sütun market cap (örneğin, "123.4\n").
* **Not**: Ekran görüntüsü al (Task\_2a\_extract.png) ve ilk satırın HTML’sini görünecek şekilde kaydet.

**Adım 2: Gerekli Kütüphaneleri ve Değişkenleri Tanımla**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Importing required libraries*

import pandas as pd

import requests

from bs4 import BeautifulSoup

*# Initialize variables*

url = "https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List\_of\_largest\_banks"

table\_attribs = ["Name", "MC\_USD\_Billion"]

log\_file = "code\_log.txt"

* **Açıklama**: requests sayfa indirmek, BeautifulSoup HTML parse etmek, pandas DataFrame için gerekli. Değişkenler proje senaryosuna uygun.

**Adım 3: log\_progress() Fonksiyonunu Tanımla**

* **Talimat**: Loglama için fonksiyon yaz (Task 1’den).
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def log\_progress(message):

'''This function logs the mentioned message of a given stage of the code execution to a log file.'''

timestamp\_format = '%Y-%m-%d-%H:%M:%S'

now = datetime.now()

timestamp = now.strftime(timestamp\_format)

with open(log\_file, "a") as f:

f.write(timestamp + ' : ' + message + '\n')

* **Açıklama**: Mesajları code\_log.txt’e <timestamp> : <message> formatında kaydeder.

**Adım 4: extract() Fonksiyonunu Yaz**

* **Talimat**: Tabloyu çekip DataFrame’e dönüştür.
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def extract(url, table\_attribs):

'''This function extracts table data from the URL and returns a Pandas DataFrame.'''

html\_page = requests.get(url).text

soup = BeautifulSoup(html\_page, 'html.parser')

df = pd.DataFrame(columns=table\_attribs)

tables = soup.find\_all('table')

for table in tables:

if "By market capitalization" in table.find\_previous('span').text:

rows = table.find\_all('tr')

for row in rows[1:]: *# Başlık satırını atla*

cols = row.find\_all('td')

if len(cols) > 1:

bank\_name = cols[1].find('a').text

market\_cap = float(cols[2].text.strip()[:-1]) *# '\n' kaldır, float'a çevir*

data\_dict = {"Name": bank\_name, "MC\_USD\_Billion": market\_cap}

df1 = pd.DataFrame(data\_dict, index=[0])

df = pd.concat([df, df1], ignore\_index=True)

break

return df

* **Açıklama**:
  + requests.get() ile sayfa indirilir.
  + BeautifulSoup ile HTML parse edilir.
  + <table> etiketleri arasında "By market capitalization" başlığını bulur.
  + <tr> ve <td> ile satır/sütunlar işlenir.
  + Market Cap’ten \n kaldırılır (strip()[:-1]), float’a çevrilir.
  + DataFrame’e eklenir.

**Adım 5: Fonksiyonu Çağır ve Çıktıyı Yazdır**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Log and extract data*

log\_progress("Data extraction complete. Initiating Transformation process")

df = extract(url, table\_attribs)

print(df)

* **Açıklama**: extract() çalıştırılır, DataFrame yazdırılır. Log dosyasına giriş yapılır.

**Adım 6: Kodu Çalıştır**

* **Komut**: python3.11 banks\_project.py
* **Beklenen Çıktı**: Terminalde DataFrame (örneğin, Name ve MC\_USD\_Billion sütunları), code\_log.txt’e log kaydı.
* **Not**: Ekran görüntüsü al (Task\_2b\_extract.png kod için, Task\_2c\_extract.png çıktı için).

**Index ve find('a') Açıklamaları**

**Tarih**: 20 Haziran 2025, Saat: 23:25 +03  
**Konu**: Pandas DataFrame ve Web Scraping Detayları  
**Kaynak**: IBM Data Engineering Kursu, ETL ve Web Scraping Lab

**1. index = [0] Neden Kullanıldı?**

* **Amaç**: df1 = pd.DataFrame(data\_dict, index=[0]) ile her satır için tek bir indeks (0) atanır.
* **Neden?**:
  + Tek satırlık veri eklerken tutarlılık sağlar.
  + pd.concat() ile birleştirildiğinde ignore\_index=True ile sıralı indeksler (0, 1, 2, ...) oluşur.
  + Çakışmayı önler, her ekleme ayrı bir satır olarak tanınır.
* **Örnek**:
* data\_dict = {"Name": "Bank X", "MC\_USD\_Billion": 123.4}
* df1 = pd.DataFrame(data\_dict, index=[0])
* # Çıktı: Indeks 0 ile tek satır
* **Bağlantı**: Banka lab’ında (Task 2) her banka için bu yöntem kullanıldı.

**2. find('a') Neden Kullanılmadı?**

* **Amaç**: market\_cap = float(cols[2].text.strip()[:-1]) ile market cap metni doğrudan işlenir.
* **Neden?**:
  + cols[2] market cap sütunu, <a> etiketi içermez (sadece metin, örneğin "123.4\n").
  + find('a') banka adları (cols[1]) için kullanılır, çünkü isimler link içinde.
  + strip()[:-1] ile \n kaldırılır, float() ile sayıya çevrilir.
* **Örnek**:
* cols[2].text # "123.4\n"
* market\_cap = float(cols[2].text.strip()[:-1]) # 123.4
* **Bağlantı**: Banka lab’ında cols[1].find('a').text isim için, cols[2].text.strip()[:-1] market cap için kullanıldı.

**İpuçları**

* **Test**: Küçük bir HTML ile index=[0] ve strip()[:-1]’i dene.
* **Not Tutma**: Notion’a bu açıklamaları ekle.

**Kaynaklar**

* **Ders**: IBM Data Engineering, ETL ve Web Scraping.
* **Dokümantasyon**: Pandas, BeautifulSoup

**Transformation of Data - Adım Adım**

**Hedef**

* DataFrame’i dönüştürerek Market Capitalization (MC) değerlerini GBP, EUR ve INR cinsinden ekle.
* Değerler, verilen exchange rate CSV dosyasına göre hesaplanmalı ve 2 ondalık basamağa yuvarlanmalı.
* Fonksiyonu çağırıp çıktıyı yazdır.
* Log kaydı yap.

**Adım 1: Gereksinimleri Anla**

* **Talimat**: Exchange rate bilgisi exchange\_rate.csv dosyasından alınacak. DataFrame’e MC\_GBP\_Billion, MC\_EUR\_Billion, MC\_INR\_Billion sütunları eklenecek.
* **Veri**: exchange\_rate.csv içinde para birimleri ve oranlar (örneğin, GBP, EUR, INR için oranlar).
* **Çıktı**: Mevcut DataFrame’e yeni sütunlar eklenecek, her değer MC\_USD\_Billion ile çarpılıp yuvarlanacak.
* **Not**: Banka lab’ından (Task 2) gelen DataFrame’in MC\_USD\_Billion sütunu kullanılacak.

**Adım 2: Gerekli Kütüphaneleri ve Değişkenleri Tanımla**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Importing required libraries*

import pandas as pd

import requests

from bs4 import BeautifulSoup

import numpy as np

from datetime import datetime

*# Initialize variables*

url = "https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List\_of\_largest\_banks"

exchange\_rate\_csv = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/exchange\_rate.csv"

table\_attribs = ["Name", "MC\_USD\_Billion"]

log\_file = "code\_log.txt"

* **Açıklama**: numpy yuvarlama için, pandas veri işleme için gerekli. exchange\_rate\_csv değişkeni oranları içerecek.

**Adım 3: log\_progress() Fonksiyonunu Tanımla**

* **Talimat**: Loglama için fonksiyon (Task 1’den).
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def log\_progress(message):

'''This function logs the mentioned message of a given stage of the code execution to a log file.'''

timestamp\_format = '%Y-%m-%d-%H:%M:%S'

now = datetime.now()

timestamp = now.strftime(timestamp\_format)

with open(log\_file, "a") as f:

f.write(timestamp + ' : ' + message + '\n')

* **Açıklama**: Log mesajlarını code\_log.txt’e kaydeder.

**Adım 4: extract() Fonksiyonunu Tanımla (Önceki Adımdan)**

* **Kod** (kısaltılmış):

python

DaraltMetni gizleÇalıştır

Kopyala

def extract(url, table\_attribs):

'''This function extracts table data from the URL and returns a Pandas DataFrame.'''

html\_page = requests.get(url).text

soup = BeautifulSoup(html\_page, 'html.parser')

df = pd.DataFrame(columns=table\_attribs)

tables = soup.find\_all('table')

for table in tables:

if "By market capitalization" in table.find\_previous('span').text:

rows = table.find\_all('tr')

for row in rows[1:]:

cols = row.find\_all('td')

if len(cols) > 1:

bank\_name = cols[1].find('a').text

market\_cap = float(cols[2].text.strip()[:-1])

data\_dict = {"Name": bank\_name, "MC\_USD\_Billion": market\_cap}

df1 = pd.DataFrame(data\_dict, index=[0])

df = pd.concat([df, df1], ignore\_index=True)

break

return df

* **Açıklama**: Task 2’den gelen fonksiyon, giriş DataFrame’ini sağlar.

**Adım 5: transform() Fonksiyonunu Yaz**

* **Talimat**: Exchange rate CSV’sini oku, yeni sütunlar ekle.
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def transform(df, csv\_path):

'''This function transforms the DataFrame by adding GBP, EUR, INR columns based on exchange rates.'''

exchange\_rate\_df = pd.read\_csv(csv\_path)

exchange\_rate = dict(zip(exchange\_rate\_df['Currency'], exchange\_rate\_df['Rate']))

df['MC\_GBP\_Billion'] = [np.round(x \* exchange\_rate['GBP'], 2) for x in df['MC\_USD\_Billion']]

df['MC\_EUR\_Billion'] = [np.round(x \* exchange\_rate['EUR'], 2) for x in df['MC\_USD\_Billion']]

df['MC\_INR\_Billion'] = [np.round(x \* exchange\_rate['INR'], 2) for x in df['MC\_USD\_Billion']]

return df

* **Açıklama**:
  + pd.read\_csv() ile exchange rate CSV’si okunur.
  + dict(zip(...)) ile para birimleri ve oranlar sözlüğe çevrilir.
  + List comprehension ile her MC\_USD\_Billion değeri ilgili oranla çarpılır ve np.round(x, 2) ile 2 ondalık basamağa yuvarlanır.
  + Yeni sütunlar DataFrame’e eklenir.

**Adım 6: Fonksiyonu Çağır ve Çıktıyı Yazdır**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Extract and transform data*

log\_progress("Data extraction complete. Initiating Transformation process")

df = extract(url, table\_attribs)

log\_progress("Data transformation complete. Initiating Loading process")

df = transform(df, exchange\_rate\_csv)

print(df)

* **Açıklama**: extract() ile veri çekilir, transform() ile dönüştürülür, DataFrame yazdırılır. Log dosyasına iki giriş yapılır.

**Adım 7: Kodu Çalıştır**

* **Komut**: python3.11 banks\_project.py
* **Beklenen Çıktı**: Terminalde DataFrame (5 sütun: Name, MC\_USD\_Billion, MC\_GBP\_Billion, MC\_EUR\_Billion, MC\_INR\_Billion), code\_log.txt’te loglar.
* **Not**: exchange\_rate.csv dosyasını indirdiğinden emin ol (wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/exchange\_rate.csv).

**Loading to CSV - Adım Adım**

**Hedef**

* Dönüştürülmüş DataFrame’i bir CSV dosyasına kaydet.
* Fonksiyonu çağırıp log kaydı yap.
* Çıktıyı doğrula.

**Adım 1: Gereksinimleri Anla**

* **Talimat**: load\_to\_csv() fonksiyonu, DataFrame’i ./Largest\_banks\_data.csv yoluna kaydedecek.
* **Veri**: Task 3’ten gelen DataFrame (5 sütun: Name, MC\_USD\_Billion, MC\_GBP\_Billion, MC\_EUR\_Billion, MC\_INR\_Billion).
* **Çıktı**: CSV dosyası oluşturulacak, log dosyasına giriş yapılacak.
* **Not**: İndeks sütunu dosyaya eklenmemeli.

**Adım 2: Gerekli Kütüphaneleri ve Değişkenleri Tanımla**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Importing required libraries*

import pandas as pd

import requests

from bs4 import BeautifulSoup

import numpy as np

from datetime import datetime

*# Initialize variables*

url = "https://web.archive.org/web/20230908091635/https://en.wikipedia.org/wiki/List\_of\_largest\_banks"

exchange\_rate\_csv = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/exchange\_rate.csv"

output\_csv\_path = "./Largest\_banks\_data.csv"

table\_attribs = ["Name", "MC\_USD\_Billion"]

log\_file = "code\_log.txt"

* **Açıklama**: pandas CSV yazımı için gerekli. output\_csv\_path proje senaryosuna uygun.

**Adım 3: log\_progress() Fonksiyonunu Tanımla**

* **Talimat**: Loglama için fonksiyon (Task 1’den).
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def log\_progress(message):

'''This function logs the mentioned message of a given stage of the code execution to a log file.'''

timestamp\_format = '%Y-%m-%d-%H:%M:%S'

now = datetime.now()

timestamp = now.strftime(timestamp\_format)

with open(log\_file, "a") as f:

f.write(timestamp + ' : ' + message + '\n')

* **Açıklama**: Log mesajlarını code\_log.txt’e kaydeder.

**Adım 4: extract() ve transform() Fonksiyonlarını Tanımla (Önceki Adımlardan)**

* **Kod** (kısaltılmış):

python

DaraltMetni gizleÇalıştır

Kopyala

def extract(url, table\_attribs):

'''This function extracts table data from the URL and returns a Pandas DataFrame.'''

html\_page = requests.get(url).text

soup = BeautifulSoup(html\_page, 'html.parser')

df = pd.DataFrame(columns=table\_attribs)

tables = soup.find\_all('table')

for table in tables:

if "By market capitalization" in table.find\_previous('span').text:

rows = table.find\_all('tr')

for row in rows[1:]:

cols = row.find\_all('td')

if len(cols) > 1:

bank\_name = cols[1].find('a').text

market\_cap = float(cols[2].text.strip()[:-1])

data\_dict = {"Name": bank\_name, "MC\_USD\_Billion": market\_cap}

df1 = pd.DataFrame(data\_dict, index=[0])

df = pd.concat([df, df1], ignore\_index=True)

break

return df

def transform(df, csv\_path):

'''This function transforms the DataFrame by adding GBP, EUR, INR columns based on exchange rates.'''

exchange\_rate\_df = pd.read\_csv(csv\_path)

exchange\_rate = dict(zip(exchange\_rate\_df['Currency'], exchange\_rate\_df['Rate']))

df['MC\_GBP\_Billion'] = [np.round(x \* exchange\_rate['GBP'], 2) for x in df['MC\_USD\_Billion']]

df['MC\_EUR\_Billion'] = [np.round(x \* exchange\_rate['EUR'], 2) for x in df['MC\_USD\_Billion']]

df['MC\_INR\_Billion'] = [np.round(x \* exchange\_rate['INR'], 2) for x in df['MC\_USD\_Billion']]

return df

* **Açıklama**: Task 2 ve 3’ten gelen fonksiyonlar, giriş DataFrame’ini sağlar.

**Adım 5: load\_to\_csv() Fonksiyonunu Yaz**

* **Talimat**: DataFrame’i CSV’ye kaydet.
* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

def load\_to\_csv(df, output\_path):

'''This function saves the final DataFrame as a CSV file in the provided path.'''

df.to\_csv(output\_path, index=False)

* **Açıklama**:
  + df.to\_csv() DataFrame’i CSV’ye yazar.
  + index=False ile indeks sütunu eklenmez.
  + output\_path ./Largest\_banks\_data.csv olarak tanımlı.

**Adım 6: Fonksiyonu Çağır ve Log Kaydı Yap**

* **Kod**:

python

DaraltMetni gizleÇalıştır

Kopyala

*# Extract, transform, and load to CSV*

log\_progress("Data extraction complete. Initiating Transformation process")

df = extract(url, table\_attribs)

log\_progress("Data transformation complete. Initiating Loading process")

df = transform(df, exchange\_rate\_csv)

log\_progress("Data saved to CSV file")

load\_to\_csv(df, output\_csv\_path)

print("Data saved to", output\_csv\_path)

* **Açıklama**: extract() ve transform() ile veri hazırlanır, load\_to\_csv() ile CSV’ye kaydedilir. Log dosyasına üç giriş yapılır.

**Adım 7: Kodu Çalıştır**

* **Komut**: python3.11 banks\_project.py
* **Beklenen Çıktı**: Terminalde "Data saved to ./Largest\_banks\_data.csv" mesajı, Largest\_banks\_data.csv dosyası oluşturulur, code\_log.txt’te loglar.
* **Doğrulama**: ./Largest\_banks\_data.csv dosyasını açıp 5 sütunun (Name, MC\_USD\_Billion, MC\_GBP\_Billion, MC\_EUR\_Billion, MC\_INR\_Billion) doğru kaydedildiğini kontrol et.

**index=False Neden Kullanıldı? - Açıklamalar**

**Tarih**: 20 Haziran 2025, Saat: 23:35 +03  
**Konu**: CSV’ye Veri Yükleme Detayları  
**Kaynak**: IBM Data Engineering Kursu, ETL Lab

**1. Açıklama**

* **Amaç**: df.to\_csv(output\_path, index=False) ile Pandas, CSV’ye indeks sütunu eklemez.
* **Neden?**:
  + İndeks (0, 1, 2, ...) veri analizi için anlamlı değil, sadece satır numaraları.
  + CSV’de gereksiz bir sütun olmasını önler, okunabilirliği artırır.
* **Sonuç**: Sadece veri sütunları (Name, MC\_USD\_Billion, vb.) kaydedilir.

**2. Örnek**

* **Kod**:
* df = pd.DataFrame({"Name": ["Bank A", "Bank B"], "MC\_USD\_Billion": [100, 200]})
* df.to\_csv("test.csv", index=False)
* **Çıktı (test.csv)**:
* Name,MC\_USD\_Billion
* Bank A,100
* Bank B,200
* **Fark (index=True ile)**:
* ,Name,MC\_USD\_Billion
* 0,Bank A,100
* 1,Bank B,200

**3. Bağlantılar**

* **Banka Lab’ı (Task 4)**: ./Largest\_banks\_data.csv’e indeks eklenmedi, veri temiz kaldı.

**4. İpuçları**

* **Test**: Küçük bir DataFrame ile index=True ve index=False’ı karşılaştır.
* **Not Tutma**: Notion’a bu açıklamayı ekle.

**5. Kaynaklar**

* **Ders**: IBM Data Engineering, ETL Lab.
* **Dokümantasyon**: Pandas to\_csv