

Institutes Info Extraction and Merging

Code Sections

Cell 1

```
!apt-get install libxml2-dev libxslt-dev python-dev
!pip install lxml
```

Cell 2

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
```

Cell 3

```
df = pd.read_csv('example.csv').reset_index(drop=True)
df
```

Cell 4

```
print(df.columns)
```

Cell 5

```
code_list = df[" code"].unique().tolist()
```

Cell 6

```
len( code_list)
```

Cell 7

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
# Now import lxml
import lxml
```

Cell 8

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Dictionary of branch codes and their corresponding codes
code_dict = {
    ##BDS
    119: [2101, 2102, 2104, 2105, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2123, 2124,
```

2127, 2134, 2135, 2207, 2211, 2212, 2229, 2230, 2313, 2314, 2325, 2326, 2331, 2332, 2333],
#BUMS
134:[5101, 5102, 5103, 5104, 5106, 5109, 5307]
#BPTH:
125:[6101, 6102, 6103, 6104, 6105, 6116, 6117, 6118, 6119, 6120, 6121, 6123, 6124, 6126, 6127, 6128, 6129, 6133, 6136, 6138, 6140, 6141, 6145, 6147, 6148, 6149, 6151, 6152, 6153, 6155, 6157, 6158, 6159, 6160, 6162, 6163, 6169, 6171, 6174, 6176, 6177, 6178, 6183, 6184, 6187, 6188, 6189, 6190, 6191, 6192, 6208, 6210, 6234, 6237, 6239, 6242, 6246, 6254, 6256, 6261, 6280, 6281, 6285, 6286, 6293, 6325, 6331, 6332, 6335, 6343, 6344, 6350, 6364, 6365, 6366, 6367, 6368, 6370, 6372, 6373, 6375, 6379, 6382
]
#BP&O
127:[9101]
#BOTH
124:[7101, 7102, 7103, 7205]
#BHMS
133:[4101, 4102, 4103, 4104, 4105, 4106, 4107, 4108, 4109, 4110, 4112, 4114, 4115, 4116, 4117, 4118, 4119, 4120, 4121, 4122, 4123, 4141, 4142, 4143, 4144, 4145, 4147, 4148, 4149, 4152, 4154, 4155, 4157, 4158, 4161, 4164, 4165, 4224, 4226, 4227, 4228, 4230, 4231, 4262, 4266, 4333, 4334, 4335, 4336, 4337, 4338, 4339, 4346, 4350, 4356, 4360, 4363]
#BAMS:
132:[3101, 3102, 3103, 3104, 3105, 3106, 3107, 3108, 3109, 3110, 3111, 3112, 3113, 3114, 3115, 3116, 3117, 3118, 3119, 3120, 3121, 3122, 3123, 3124, 3125, 3126, 3128, 3129, 3130, 3131, 3132, 3133, 3134, 3135, 3136, 3137, 3138, 3139, 3140, 3141, 3142, 3147, 3158, 3160, 3161, 3162, 3172, 3176, 3179, 3180, 3181, 3182, 3183, 3184, 3185, 3187, 3188, 3191, 3192, 3193, 3196, 3198, 3199, 3232, 3233, 3234, 3235, 3236, 3237, 3238, 3239, 3240, 3241, 3252, 3257, 3263, 3264, 3265, 3266, 3271, 3273, 3274, 3277, 3286, 3294, 3295, 3296, 3342, 3343, 3344, 3353, 3354, 3355, 3367, 3368, 3369, 3375, 3378, 3389, 3390, 3395, 3396, 3397, 3398, 3399]
#BASLP
126:[8101, 8102]
#MBBS
118:[1101, 1102, 1103, 1104, 1105, 1108, 1109, 1110, 1112, 1114, 1115, 1118, 1119, 1120, 1132, 1135, 1136, 1137, 1138, 1139, 1140, 1143, 1144, 1147, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1159, 1221, 1222, 1223, 1225, 1226, 1234, 1241, 1242, 1248, 1261, 1327, 1328, 1329, 1330, 1333, 1345, 1358, 1360, 1362]
}

```

def fetch_data(branch_code):
    headers_printed = False # Flag to track if headers have been printed

    if branch_code in code_dict:
        for code in code_dict[branch_code]:
            combined_code = f"{branch_code}{code}"
            url = f'https://cetcell.mahacet.org/search-institute-
deatils/?getinstitute={combined_code}'
            response = requests.get(url)
            if response.status_code == 200:
                page_content = response.text
                soup = BeautifulSoup(page_content, 'html.parser')
                lt = soup.find_all('table')
                if lt:
                    ci = str(lt[0])
                    df = pd.read_html(str(lt[1]))[0]
                    if not headers_printed:
                        print(df.columns)
                        headers_printed = True
                    print(df)
                else:
                    print(f"No tables found for Combined Code: {combined_code}")
            else:
                print(f"Failed to retrieve data for Combined Code: {combined_code}, Status Code:
{response.status_code}")
        else:
            print(f"Branch code {branch_code} not found in the dictionary.")

# Example usage
branch_code_input = 94 # Change this to the desired branch code
fetch_data(branch_code_input)

```

Cell 9

```

def decode_cf_email(encoded_string):
    decoded = ""
    k = int(encoded_string[:2], 16)
    for i in range(2, len(encoded_string)-1, 2):
        decoded += chr(int(encoded_string[i:i+2], 16) ^ k)
    return decoded

def fetch_data(branch_code):
    merged_df_list = []
    headers_printed = False # Flag to track if headers have been printed

```

```

if branch_code in code_dict:
    for code in code_dict[branch_code]:
        combined_code = f"{branch_code}{code}"
        url = f'https://cetcell.mahacet.org/search-institute-
deatils/?getinstitute={combined_code}'
        response = requests.get(url)
        if response.status_code == 200:
            page_content = response.text
            soup = BeautifulSoup(page_content, 'html.parser')
            lt = soup.find_all('table')
            # Extract and decode emails
            encoded_emails = soup.find_all('a', {'class': '__cf_email__'})
            decoded_emails = []
            for encoded_email in encoded_emails:
                data_cfemail = encoded_email['data-cfemail']
                decoded_email = decode_cf_email(data_cfemail)
                decoded_emails.append(decoded_email)

            if not lt:
                print(f"No tables found for Combined Code: {combined_code}")
                continue

            ci = str(lt[0])
            try:
                ci_df = pd.read_html(ci)[0]
            except Exception as e:
                print(f"Error parsing table for Combined Code: {combined_code}, Error: {e}")
                result = pd.DataFrame([{'Department Name': np.nan, 'Institute Name': np.nan, 'District': np.nan, 'City':
np.nan,
'University': np.nan, 'Institute Status': np.nan, 'Minority Status': np.nan,
'E-Mail ID': np.nan, 'College Code': combined_code, 'Address': np.nan, 'Taluka':
np.nan,
'PIN Code': np.nan, 'Establishment Year': np.nan, 'Autonomy Status': np.nan,
'Phone Number': np.nan, 'Website URL': np.nan, 'Course Name': np.nan,
'Course Type': np.nan, 'Branch Name': np.nan, 'Sanction Intake': np.nan
}])
                merged_df_list.append(result)
                print(f'{combined_code} done unsuccessfully')
                continue

            data = []

```

```

i = 0
row = {}
while i < 4:
    for j in range(len(ci_df[i].tolist())):
        row[f"{ci_df[i].tolist()[j]}"] = ci_df[i + 1].tolist()[j]
    i += 2
data.append(row)
df1 = pd.DataFrame(data)
# Add decoded emails to df1
if decoded_emails:
    df1["E-Mail ID"] = decoded_emails[0] # Assuming one email per institute
if len(df1.columns) != 16:
    print(f'for {combined_code} number of columns is {len(df1.columns)}')

df2 = pd.read_html(str(lt[1]))[0]
result = pd.concat([df2] * len(df1)).reset_index(drop=True)
for col in df1.columns:
    result[col] = df1.iloc[0][col]
result = result[df1.columns.tolist() + df2.columns.tolist()]
result = result.rename(columns={"Sub Course Name": "Branch Name", "Institute
code": "College Code"})
result["College Code"] = result["College Code"].astype(float)
merged_df_list.append(result)
print(f'{combined_code} done')
else:
    print(f'Failed to retrieve data for Combined Code: {combined_code}, Status Code:
{response.status_code}')
else:
    print(f'Branch code {branch_code} not found in the dictionary.')

# Concatenate all dataframes in merged_df_list into a single dataframe
if merged_df_list:
    final_df = pd.concat(merged_df_list, ignore_index=True)
    return final_df
else:
    return pd.DataFrame() # Return an empty dataframe if no data was collected

# Example usage
branch_code_input = 134 # Change this to the desired branch code
final_df = fetch_data(branch_code_input)

# Print the final dataframe
print(final_df)

```

Cell 10

```
len(final_df)
```

Cell 11

```
final_df.shape
```

Cell 12

```
len(final_df["College Code"].unique())
```

Cell 13

```
final_df.loc[final_df["College Code"]==6006]
```

Cell 14

```
final_df.iloc[1813:1888,:]
```

Cell 15

```
final_df["Course Type"].unique()
```

Cell 16

```
temp = final_df.loc[final_df["Course Type"]!= 'Under Graduate Courses']  
temp = temp.loc[temp["Course Type"]!= 'Post Graduate Courses']  
temp
```

Cell 17

```
final_df.isna().sum()
```

Cell 18

```
final_df
```

Cell 19

```
final_df.to_csv('coursename_collegeinfo.csv', index=False)
```

Cell 20

```
li = ['Department Name', 'Institute Name', 'District', 'City', 'University',  
      'Institute Status', 'Minority Status', 'E-Mail ID', 'College Code',  
      'Address', 'Taluka', 'PIN Code', 'Establishment Year',  
      'Autonomy Status', 'Phone Number', 'Website URL', 'Course Name',  
      'Course Type', 'Branch Name', 'Sanction Intake']
```

Cell 21

```
import pandas as pd  
df = pd.read_csv('csv file which contain cutoff data')
```

```
df2 = pd.read_csv('course_collegeinfo.csv')
```

Cell 22

```
df.columns
```

Cell 23

```
df2.columns
```

Cell 24

```
import pandas as pd
df = pd.read_csv('csv file which contain cutoff data')
df2 = pd.read_csv('course_collegeinfo.csv')

# Convert 'code' column in df to numeric type, handling potential errors
df['code'] = pd.to_numeric(df['code'], errors='coerce')

# Merge the dataframes on 'code' and 'College Code'
merged_df = pd.merge(df, df2, left_on='code', right_on='College Code', how='outer')

# Drop rows where 'College Code' is missing or null
merged_df = merged_df.dropna(subset=['College Code'])

print(merged_df)
```

Cell 25

```
merged_df.to_csv('cutoff data and course name files_merge.csv', index=False)
```

Explanation Sections

Cell 1 Explanation

This cell installs the necessary dependencies for parsing and processing XML/HTML data. The `lxml` library is a Pythonic binding for the C libraries libxml2 and libxslt, used for processing XML and HTML. The `apt-get` commands ensure that these libraries are available.

Cell 2 Explanation

This cell imports necessary Python libraries:

- `requests`: For making HTTP requests to download web content.

- `BeautifulSoup` from `bs4`: For parsing HTML and XML documents.
- `pandas`: For data manipulation and analysis.
- `numpy`: For numerical operations, often used with `pandas` for advanced data manipulation.

Cell 3 Explanation

This cell reads a CSV file named 'example.csv' into a `pandas DataFrame`, then resets the index of the `DataFrame`. The `.reset_index(drop=True)` ensures that the old index is dropped and not added as a new column.

Cell 4 Explanation

This cell prints the column names of the `DataFrame df`. It helps in understanding the structure of the dataset.

Cell 5 Explanation

This cell extracts the unique values from the column 'code' in the `DataFrame df` and stores them as a list called `code_list`. The `.unique()` method returns an array of unique values, and `.tolist()` converts it into a Python list.

Cell 6 Explanation

Length of the list is printed.

Cell 7 Explanation

Imports required libraries of warnings.

Cell 8 Explanation

This Python script extracts information from a website using HTTP requests. It uses the `requests` library to fetch HTML data and `BeautifulSoup` for parsing. The script defines a dictionary `code_dict` mapping branch codes to corresponding codes. The `fetch_data` function constructs a URL for each branch code, sends a request, and parses the returned HTML to extract table data. It then prints the column headers once and displays the data for each code in the branch. If no tables are found or the request fails, it prints an error message. The script ends by calling `fetch_data` with a specified branch code.

Cell 9 Explanation

This script fetches institute data from a website based on a given branch code. It starts by decoding any Cloudflare-protected email addresses using the `decode_cf_email` function. The `fetch_data` function then constructs URLs, sends HTTP requests, and parses the returned HTML to extract tables. It decodes email addresses, processes table data into data frames, and appends them to a list. If no tables are found or there's an error, it logs an appropriate message. Finally, the collected data frames are concatenated into a single data frame (`final_df`) and returned for further use.

Cell 10 Explanation

Prints the length of 'final_df'

Cell 11 Explanation

Defines the shape of the given dataframe

Cell 12 Explanation

Prints the length of the given structure

Cell 13 Explanation

Locates the given indices

Cell 14 Explanation

Locates the given indices

Cell 15 Explanation

Prints the number of unique values in the dataframe

Cell 16 Explanation

This code filters the 'final_df' DataFrame to remove rows where the "Course Type" is either "Under Graduate Courses" or "Post Graduate Courses." The resulting 'temp' DataFrame contains only the rows with other course types.

Cell 17 Explanation

Prints the count of null values

Cell 18 Explanation

prints the dataframe

Cell 19 Explanation

Converts the data frame into a csv

Cell 20 Explanation

defines an array of attributes/column names

Cell 21 Explanation

This code loads two CSV files into Pandas DataFrames. The first file, 'csv file which contain cutoff data', is read into 'df', and the second file, 'course_collegeinfo.csv', is read into 'df2'. These DataFrames will hold the data from their respective CSV files for further analysis or processing.

Cell 22 Explanation

Prints the column names of the data frame 'df'

Cell 23 Explanation

Prints the column names of the data frame 'df2'

Cell 24 Explanation

1. Import Libraries: You import the 'pandas' library as 'pd'.
2. Load Data: You read two CSV files into DataFrames 'df' and 'df2'.
3. Convert Data: You convert the 'code' column in 'df' to numeric values, replacing non-convertible values with 'NaN'.
4. Merge DataFrames: You merge 'df' and 'df2' using 'code' from 'df' and 'College Code' from 'df2', performing an outer join to include all rows from both DataFrames.
5. Drop Missing Values: You remove rows from 'merged_df' where 'College Code' is missing or null.
6. Print Results: You display the resulting DataFrame 'merged_df'.

The overall goal is to combine data from two sources based on a common code, while ensuring that only rows with valid college codes are kept.

Cell 25 Explanation

1. Save DataFrame to CSV: 'merged_df.to_csv('cutoff data and course name files_merge.csv', index=False)' saves the 'merged_df' DataFrame to a CSV file.
2. Specify File Name: The output file is named 'cutoff data and course name files_merge.csv'.
3. Exclude Index: The parameter 'index=False' ensures that the row indices are not included in the saved CSV file, so only the DataFrame's data and column headers are written to the file.

Input:

	code_college	category	gender	code	Cap_round_No	min_air	min_sr_no	min_neet_roll_no	min_cet_form_no	min_quota	max_air	max_sr_no	max_neet_roll_no	max_cet_form_no	max_quota
0	6105.SANCHETI PT PUNE(Ret.)	OBC	F	6105	1	263229	176	3115170446	232009178	OPEN	263229	176	3115170446	232009178	OPEN
1	6116.KJS PT,SION UBAl(Ret.)	NTC	F	6116	1	252971	160	3109070088	232013191	OPEN	252971	160	3109070088	232013191	OPEN
2	6117.VPS PT NASIK(Ret.)	OBC	F	6117	1	379939	428	3113030199	232006804	OPEN	379939	428	3113030199	232006804	OPEN
3	6118.TERANA PT NAVI UBAl	OBC	F	6118	1	386931	452	3129060241	232010141	OPEN	386931	452	3129060241	232010141	OPEN
4	6118.TERANA PT NAVI UBAl	SC	F	6118	1	380927	429	3109220308	232052178	OPEN	380927	429	3109220308	232052178	OPEN
...
7226	8102.TN BASLP UBAl(No R3 Pref)	SOBC	F	8102	3	247420	175	3112190317	232049751	OBC(W)	247420	175	3112190317	232049751	OBC(W)
7227	9101.AIIR BPO UBAl	EWS	F	9101	3	437864	650	3126040169	232035275	EWS	437864	650	3126040169	232035275	EWS
7228	9101.AIIR BPO UBAl	OBC	F	9101	3	431608	628	3132020110	232052987	OBC	431608	628	3132020110	232052987	OBC
7229	9101.AIIR BPO UBAl	SC	M	9101	3	295921	265	3128060388	232006799	SC	295921	265	3128060388	232006799	SC
7230	9101.AIIR BPO UBAl	VJA	M	9101	3	861739	2316	3114010780	232044769	VJA	861739	2316	3114010780	232044769	VJA

7231 rows x 16 columns

Output:

	code_college	category	gender	code	Cap_round_No	\
6783	5101:AE TIBIA MUMBAI	EWS	M	5101.0	1	
6784	5101:AE TIBIA MUMBAI	OBC	F	5101.0	1	
6785	5101:AE TIBIA MUMBAI(Ret.)	OBC	F	5101.0	1	
6786	5101:AE TIBIA MUMBAI	EWS	M	5101.0	1	
6787	5101:AE TIBIA MUMBAI	OBC	M	5101.0	1	
...	
6964	5307:YUMC KANNAD(No Change)	OBC	M	5307.0	3	
6965	5307:YUMC KANNAD(No Change)	I.Q.	F	5307.0	3	
6966	5307:YUMC KANNAD(Ret.)	I.Q.	M	5307.0	3	
6967	5307:YUMC KANNAD(No Change)	I.Q.	F	5307.0	3	
6968	5307:YUMC KANNAD(Ret.)	I.Q.	M	5307.0	3	
	min_air	min_sr_no	min_neet_roll_no	min_cet_form_no	min_quota	...
6783	403083	7315	3122070199	232047933	HA	...
6784	407622	7400	3135010091	232000225	OPEN	...
6785	315900	5424	3135010502	232055286	OPEN	...
6786	349404	6162	3109210123	232050507	OPEN	...
6787	357302	6334	3130010618	232003728	OPEN	...
...
6964	496672	10333	3130010174	232011331	OPEN	...
6965	838994	16906	3104050103	232030173	MINO	...
6966	905739	17779	3110060478	232013261	MINO	...
6967	838994	16906	3104050103	232030173	MINO	...
6968	905739	17779	3110060478	232013261	MINO	...
	Taluka	PIN Code	Establishment Year	Autonomy Status	Phone Number	\
6783	Andheri	400061.0	NaN	Not Applicable	9.967818e+09	
6784	Andheri	400061.0	NaN	Not Applicable	9.967818e+09	
6785	Andheri	400061.0	NaN	Not Applicable	9.967818e+09	
6786	Andheri	400061.0	NaN	Not Applicable	9.967818e+09	
6787	Andheri	400061.0	NaN	Not Applicable	9.967818e+09	
...	
6964	Kannad	431103.0	NaN	Not Applicable	9.545070e+09	
6965	Kannad	431103.0	NaN	Not Applicable	9.545070e+09	
6966	Kannad	431103.0	NaN	Not Applicable	9.545070e+09	
6967	Kannad	431103.0	NaN	Not Applicable	9.545070e+09	
6968	Kannad	431103.0	NaN	Not Applicable	9.545070e+09	
	Website URL	Course Name	Course Type	\		
6783	www.aitibbianumbai.co.in	BUMS Under Graduate Courses				
6784	www.aitibbianumbai.co.in	BUMS Under Graduate Courses				
6785	www.aitibbianumbai.co.in	BUMS Under Graduate Courses				
6786	www.aitibbianumbai.co.in	BUMS Under Graduate Courses				
6787	www.aitibbianumbai.co.in	BUMS Under Graduate Courses				
...			
6964	yfunc.in	BUMS Under Graduate Courses				
6965	yfunc.in	BUMS Under Graduate Courses				
6966	yfunc.in	BUMS Under Graduate Courses				
6967	yfunc.in	BUMS Under Graduate Courses				
6968	yfunc.in	BUMS Under Graduate Courses				
	Branch Name	Sanction Intake				
6783	BUMS	60.0				
6784	BUMS	60.0				
6785	BUMS	60.0				
6786	BUMS	60.0				
6787	BUMS	60.0				