P3_Data_Analysis_and_Web_Scraping

December 15, 2023

1 Project 4 - Data Analysis and Web Scraping

1.1 Import essential libraries

[126]:
```
import os
import shutil
import csv
import zipfile
import pandas as pd
import time
import glob
import math

import selenium
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from bs4 import BeautifulSoup

def PRINT() -> None: print('-'*80)
def PRINT(sent) -> None : print(f"{'-'*100}\n{sent}\n{'-'*100}")
```

[3]:
```
os.environ['PATH'] += r'C:
↪\Users\gavvi\ChromeDrivers\chrome-win64\chrome-win64\chrome.exe'
```

1.2 Preforming Data Scraping Techniques to Extract the Dataset

1.2.1 Web Scraping Setup

The first step is to open the chrome website and navigate to *chEMBL* database official website and enter in the *search* location `integrins` in order to get the wanted data for our mission.

[4]:
```
# Open chrome website
driver = webdriver.Chrome()

# Use the correct URL in orer to navigate to the correct dataset in chEMBL␣
↪database website
```

```
url = 'https://www.ebi.ac.uk/chembl/'
driver.get(url)

driver.implicitly_wait(5) # wait 5sec in case there are server issues
```

The next step is to type `Integrins` in the search box and press enter in order to execute the search.

[5]:
```
# search for integrins
search_input = driver.find_element("name","search-str")

# type Integrings into the search field
field = "Integrins"
search_input.send_keys(field)

# press enter to preform the search
search_input.send_keys(Keys.ENTER)
```

Next, we want to move to the `Targets` window. Thus, we need to enter on the button that will take as to the correct window e.g. press on `Targets` button

[6]:
```
target_btn = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR,␣
 ↪'[data-resource-key="TARGET"]'))
)
target_btn.click()

driver.implicitly_wait(5) # wait 5sec in case there are server issues
```

The final data filtering step we have is to take only the next types:

- PROTEIN COMPLEX
- PROTEIN COMPLEX GROUP
- PROTEIN-PROTEIN INTERACTION
- SELECTIVITY GROUP

As before, we will achive that by pressing the corresponding buttons that will filter the correct data

[7]:
```
# scroll down by 1900 pixels so that the buttons will be visible
driver.execute_script("window.scrollBy(0, 1500);")

# Wait for 5 seconds
time.sleep(5)

driver.execute_script("window.scrollBy(0, 400);")

# Wait for 5 seconds
time.sleep(5)
```

[8]:
```
filter_n1_btn = WebDriverWait(driver, 20).until(
    EC.presence_of_element_located((By.CSS_SELECTOR, '.
 ↪front-bar[data-facet-group-key="target_type"][data-facet-key="PROTEIN␣
 ↪COMPLEX"]'))
)
filter_n1_btn.click()

filter_n2_btn = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CLASS_NAME,␣
 ↪'bucket[data-bucket-key="PROTEIN COMPLEX GROUP"]'))
)
filter_n2_btn.click()

filter_n3_btn = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CLASS_NAME,␣
 ↪'bucket[data-bucket-key="PROTEIN-PROTEIN INTERACTION"]'))
)
filter_n3_btn.click()

filter_n4_btn = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.CLASS_NAME,␣
 ↪'bucket[data-bucket-key="SELECTIVITY GROUP"]'))
)
filter_n4_btn.click()
```

1.3 Manually downloading all of compounds csv files

1.4 Extraxting the csv files from the downloaded zipped files

[43]:
```
zip_folder_path = 'Integrins - common Compounds zips'
csv_folder_path = 'Integrins - common Compounds csv'
temp_dir = "csv_temp_dir"

# Get a list of all zip files
zip_files = [file for file in os.listdir(zip_folder_path) if file.endswith('.
 ↪zip')]

PRINT(zip_files)
```

```
--------------------------------------------------------------------------------
--------------------
['1.zip', '10.zip', '11.zip', '12.zip', '13.zip', '14.zip', '15.zip', '16.zip',
'17.zip', '18.zip', '19.zip', '2.zip', '20.zip', '21.zip', '22.zip', '23.zip',
'24.zip', '25.zip', '26.zip', '27.zip', '28.zip', '29.zip', '3.zip', '30.zip',
'31.zip', '32.zip', '33.zip', '34.zip', '35.zip', '4.zip', '5.zip', '6.zip',
'7.zip', '8.zip', '9.zip']
--------------------------------------------------------------------------------
--------------------
```

As we can see from the line above, creating list data structure shuffles the *.zip* files in some kind of random order.

But as mentioned before, the order in matter because we want to map for each row in main data frame, i.e. the data frame with 35 rows and contains the *Compounds* column in it, its corresponding CSV file of molecules with the same data but differend *SMILES* value.

In order to retrieve the wanted order of the *.zip* file, we will write two helper function in order to sort the list data structure by the *.zip* file name number.

[44]:
```
import re

def natural_sort_key(s):
    return [int(text) if text.isdigit() else text.lower() for text in re.
 ↪split('([0-9]+)', s)]

def natural_sort(input_list):
    return sorted(input_list, key=natural_sort_key)
```

[47]:
```
# Sort the list to keep the right order of the zip files (i.e. 1,2,3,4... and␣
 ↪not 1,9,5,10)
zip_files = natural_sort(zip_files)

for index, zip_file in enumerate(zip_files, start=1):
    zip_file_path = os.path.join(zip_folder_path, zip_file)

    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        # Extract the current zip file to the temp_dir
        zip_ref.extractall(temp_dir)

    # Get the current extracted csv file path
    curr_csv_path = glob.glob(os.path.join(temp_dir, '*.csv'))

    if len(curr_csv_path) == 1:
        curr_csv_path = curr_csv_path[0]

        # Specify the new name
        new_name = f'compounds_csv_{index}.csv'
        new_path = os.path.join(csv_folder_path, new_name)

        # Rename the file
        os.rename(curr_csv_path, new_path)

        PRINT(f'Extracted the zip file number {index} and Renamed: {zip_file}␣
 ↪-> {new_name}')
```

```
Extracted the zip file number 1 and Renamed: 1.zip -> compounds_csv_1.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 2 and Renamed: 2.zip -> compounds_csv_2.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 3 and Renamed: 3.zip -> compounds_csv_3.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 4 and Renamed: 4.zip -> compounds_csv_4.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 5 and Renamed: 5.zip -> compounds_csv_5.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 6 and Renamed: 6.zip -> compounds_csv_6.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 7 and Renamed: 7.zip -> compounds_csv_7.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 8 and Renamed: 8.zip -> compounds_csv_8.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 9 and Renamed: 9.zip -> compounds_csv_9.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 10 and Renamed: 10.zip -> compounds_csv_10.csv
--------------------------------------------------------------------------------
--------------------
```

```
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 11 and Renamed: 11.zip -> compounds_csv_11.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 12 and Renamed: 12.zip -> compounds_csv_12.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 13 and Renamed: 13.zip -> compounds_csv_13.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 14 and Renamed: 14.zip -> compounds_csv_14.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 15 and Renamed: 15.zip -> compounds_csv_15.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 16 and Renamed: 16.zip -> compounds_csv_16.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 17 and Renamed: 17.zip -> compounds_csv_17.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 18 and Renamed: 18.zip -> compounds_csv_18.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 19 and Renamed: 19.zip -> compounds_csv_19.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 20 and Renamed: 20.zip -> compounds_csv_20.csv
```

```
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 21 and Renamed: 21.zip -> compounds_csv_21.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 22 and Renamed: 22.zip -> compounds_csv_22.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 23 and Renamed: 23.zip -> compounds_csv_23.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 24 and Renamed: 24.zip -> compounds_csv_24.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 25 and Renamed: 25.zip -> compounds_csv_25.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 26 and Renamed: 26.zip -> compounds_csv_26.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 27 and Renamed: 27.zip -> compounds_csv_27.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 28 and Renamed: 28.zip -> compounds_csv_28.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 29 and Renamed: 29.zip -> compounds_csv_29.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
```

```
--------------------
Extracted the zip file number 30 and Renamed: 30.zip -> compounds_csv_30.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 31 and Renamed: 31.zip -> compounds_csv_31.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 32 and Renamed: 32.zip -> compounds_csv_32.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 33 and Renamed: 33.zip -> compounds_csv_33.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 34 and Renamed: 34.zip -> compounds_csv_34.csv
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Extracted the zip file number 35 and Renamed: 35.zip -> compounds_csv_35.csv
--------------------------------------------------------------------------------
--------------------
```

Next, we can delete the empty *.zip* files

[48]:
```python
for zip_file in zip_files:
    zip_file_path = os.path.join(zip_folder_path, zip_file)
    os.remove(zip_file_path)
```

Next, we will visualize random `csv` file from "*Integrins - common Compounds csv*" directory

[49]:
```python
# Get the csv directory path
csv_directory_path = 'Integrins - common Compounds csv'

# Get the first csv file path
first_csv_path = os.path.join(csv_directory_path, 'compounds_csv_1.csv')

df = pd.read_csv(first_csv_path)

df.head(5)
```

[49]:
```
  ChEMBL ID;"Name";"Synonyms";"Type";"Max Phase";"Molecular
Weight";"Targets";"Bioactivities";"AlogP";"Polar Surface Area";"HBA";"HBD";"#RO5
Violations";"#Rotatable Bonds";"Passes Ro3";"QED Weighted";"CX Acidic pKa";"CX
Basic pKa";"CX LogP";"CX LogD";"Aromatic Rings";"Structure Type";"Inorganic
Flag";"Heavy Atoms";"HBA (Lipinski)";"HBD (Lipinski)";"#RO5 Violations
(Lipinski)";"Molecular Weight (Monoisotopic)";"Np Likeness Score";"Molecular
Species";"Molecular Formula";"Smiles";"Inchi Key"
0  CHEMBL2069439;"";"";"Unknown";"";"2532.95";"1"…
1  CHEMBL2069367;"";"";"Unknown";"";"2047.40";"1"…
2  CHEMBL2069368;"";"";"Unknown";"";"2027.42";"1"…
3  CHEMBL2069370;"";"";"Unknown";"";"2006.44";"1"…
4  CHEMBL2069363;"";"";"Unknown";"";"2002.41";"1"…
```

From the data frame, it seems we need to specify the delimiter ; in order to get the table with the data in more readable way

[50]:
```
df = pd.read_csv(first_csv_path, sep=';')

df.head(5)
```

[50]:

	ChEMBL ID	Name	Synonyms	Type	Max Phase	Molecular Weight	\
0	CHEMBL2069439	NaN	NaN	Unknown	NaN	2532.95	
1	CHEMBL2069367	NaN	NaN	Unknown	NaN	2047.40	
2	CHEMBL2069368	NaN	NaN	Unknown	NaN	2027.42	
3	CHEMBL2069370	NaN	NaN	Unknown	NaN	2006.44	
4	CHEMBL2069363	NaN	NaN	Unknown	NaN	2002.41	

	Targets	Bioactivities	AlogP	Polar Surface Area	…	Heavy Atoms	\
0	1	2	NaN	NaN	…	NaN	
1	1	2	NaN	NaN	…	NaN	
2	1	2	NaN	NaN	…	NaN	
3	1	2	NaN	NaN	…	NaN	
4	1	5	NaN	NaN	…	NaN	

	HBA (Lipinski)	HBD (Lipinski)	#RO5 Violations (Lipinski)	\
0	NaN	NaN	NaN	
1	NaN	NaN	NaN	
2	NaN	NaN	NaN	
3	NaN	NaN	NaN	
4	NaN	NaN	NaN	

	Molecular Weight (Monoisotopic)	Np Likeness Score	Molecular Species	\
0	2531.3966	NaN	NaN	
1	2046.1255	NaN	NaN	
2	2026.1357	NaN	NaN	
3	2005.1718	NaN	NaN	
4	2001.1405	NaN	NaN	

```
   Molecular Formula                                                Smiles  \
0    C103H178N50O24S  N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…
1     C84H143N41O18S  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
2     C85H143N41O16S  N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…
3     C84H148N40O16S  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
4     C84H144N40O16S  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…

                     Inchi Key
0  IWGVTNFOZGEWIP-YOPUZRBESA-N
1  VTAUSMPSIPZBTN-ZJRJYKPTSA-N
2  FXABLJUCIVJEAN-RJMYSKKESA-N
3  FZHZWQOBTKVEPC-ZJRJYKPTSA-N
4  JSDLSOZYYZIPPO-ZJRJYKPTSA-N

[5 rows x 33 columns]
```

From all of the columns in the data frame we got, we want to keep only the `SMILES` column in order to preforme *join* later on with the data frame we got from the previous project (i.e. project 3) in order to merge the new data (i.e. target pref names, uniprot ids (1 & 2) and organism name)

1.5 Reading and Dropping Data from CSV Files

The next steps include:

- Reading the csv file into data frames with specification of delimiter equal to ';'
- Dropping all of the columns except the column which contains the molecules SMILES in each and every data frame

[51]:
```
source_directory_path = 'Integrins - common Compounds csv'
target_directory_path = "Integrins - common Compounds csv - SMILES only"
```

[52]:
```
!pwd
```

```
/cygdrive/c/Users/gavvi/Desktop/Programming/GitHub/DeepLearningResearchStarship/
Project 3 Web Scraping and Data Analysis
```

[53]:
```
for indx, csv in enumerate(natural_sort(os.listdir(source_directory_path)),␣
↪start=1):

    # Read the csv file into data frame with specification of the correct␣
↪delimiter (i.e. ';')
    curr_df = pd.read_csv(os.path.join(source_directory_path, csv), sep=';')

    # Keep only the column named 'Smiles', and drop all of the other columns
    curr_df.drop(curr_df.columns.difference(['Smiles']), axis=1, inplace=True)

    # Save the currect format csv file to target directory
    curr_df.to_csv(os.path.join(target_directory_path,␣
↪f'compounds_SMILES_{indx}.csv'), index=False)
```

1.6 Extract the Main Integrins Table and Merge with Privious CSVs

The corresponding step involves extracting the main Integrins table, which includes data such as the name (i.e., the target preferred name that we want to extract), UniProts 1 and 2 (which can be multiple), organism, and merging that with the CSV files.

The merging step: Each row in the main data frame corresponds to a CSV file containing *SMILES* (i.e. simplified molecular-input line-entry system) values for that row. This implies that each row in the main data frame will appear a number of times equal to the number of rows in its corresponding CSV file. The only data that will change is the unique SMILES value for that current row.

1.6.1 Open chEMBL Database Site

First, we need to reopen the *chEMBL* database internet site and navigate to the correct *Integrins* filtered table as in the steps before. In order to achive that, we can just rerun the cells above that navigate us to the currect point and filter for us all of the wanted data using *selenium* scraping library.

1.6.2 Extract the Main Data Frame Table

We can easily extract the data frame by pressing on the *csv* button and then on the *here* button in order to download the csv file.

Note that we dont need to preforme data scraping steps because manually downloading is much easier and contains only few simple steps.

1.6.3 Merge Step

Finally, we can merge the main data frame with all of the other data frames which contains the *SMILES* of the molecules as mentioned before.

[169]:
```
main_csv = "Integrins_main_data_frame.csv"
compounds_SMILES_dir = "Integrins - common Compounds csv - SMILES only"
```

[170]:
```
df = pd.read_csv(main_csv, sep=';')

df.head(10)
```

[170]:

	ChEMBL ID	Name
0	CHEMBL3430893	Integrin alpha-V/alpha-5
1	CHEMBL3137268	Integrin alpha2/beta1
2	CHEMBL3137286	EZH2/SUZ12/EED complex
3	CHEMBL2111481	Integrin alpha-4/beta-7
4	CHEMBL4106121	MAC1-CD40L
5	CHEMBL3137278	Integrin alpha1/beta1 complex
6	CHEMBL3883284	Integrin alpha-3/beta-3
7	CHEMBL2111443	Integrin alpha-V/beta-3 and alpha-IIb/beta 3
8	CHEMBL4748218	VHL/Polycomb protein EED

```
9  CHEMBL3301388                    EZH2/SUZ12/EED/RBBP7/RBBP4

                  UniProt Accessions                         Type  \
0                       P06756|P08648             PROTEIN COMPLEX
1                       P05556|P17301             PROTEIN COMPLEX
2                  Q15910|O75530|Q15022             PROTEIN COMPLEX
3                       Q00651|P26011             PROTEIN COMPLEX
4                  P05107|P11215|P29965  PROTEIN-PROTEIN INTERACTION
5                       P05556|P56199             PROTEIN COMPLEX
6                       P05106|P26006             PROTEIN COMPLEX
7                  P05106|P06756|P08514           SELECTIVITY GROUP
8                       O75530|P40337  PROTEIN-PROTEIN INTERACTION
9  Q15910|O75530|Q15022|Q09028|Q16576             PROTEIN COMPLEX

       Organism  Compounds  Activities  Tax ID  Species Group Flag
0  Homo sapiens          6           6    9606               False
1  Homo sapiens         57          62    9606               False
2  Homo sapiens         10          16    9606               False
3  Mus musculus         22          55   10090               False
4  Homo sapiens          6           6    9606               False
5  Homo sapiens          6           6    9606               False
6  Homo sapiens         25          47    9606               False
7  Homo sapiens         49          49    9606               False
8  Homo sapiens          2           6    9606               False
9  Homo sapiens          7          11    9606               False
```

We have observed an issue regarding the sequence of lines in the main data frame. The arrangement of the rows does not align with the order on the chEMBL site. This misalignment may stem from a hiccup during the website download step, causing a reshuffling of the rows.

As a result, we must rearrange the rows to match the sequence on the chEMBL website. This adjustment is crucial because all SMILES CSV files are organized in correlation with the rows in the main data frame. To ensure a seamless merge, it is imperative to reorder the rows in the accurate sequence.

Reordering the Rows of the Main Data Frame

[171]:
```
cmEMBL_right_order = ['CHEMBL2111481', 'CHEMBL2095184', 'CHEMBL3430894',␣
 ↪'CHEMBL3430895', 'CHEMBL3137278',
                       'CHEMBL2095226', 'CHEMBL1907599' , 'CHEMBL3885597',␣
 ↪'CHEMBL3137268', 'CHEMBL3883284',
                       'CHEMBL2093869', 'CHEMBL3885596', 'CHEMBL2111461',␣
 ↪'CHEMBL2096675', 'CHEMBL3430893',
                       'CHEMBL2111416', 'CHEMBL2111425', 'CHEMBL4106150',␣
 ↪'CHEMBL2111362', 'CHEMBL2364172',
                       'CHEMBL2111443','CHEMBL3430891', 'CHEMBL2111407',␣
 ↪'CHEMBL1907598', 'CHEMBL2096661',
```

```
                    'CHEMBL4106121', 'CHEMBL3885595', 'CHEMBL4523628',␣
 ↪'CHEMBL3430892', 'CHEMBL4296069',
                    'CHEMBL3883325', 'CHEMBL4748218', 'CHEMBL3137286',␣
 ↪'CHEMBL3137287', 'CHEMBL3301388']
```

[201]:
```
# Verify we got all of the 35 values

PRINT(f'The number of unique chEMBL ids are 35, and we got ->␣
 ↪{len(cmEMBL_right_order)}')
```

```
--------------------------------------------------------------------------------
--------------------
The number of unique chEMBL ids are 35, and we got -> 35
--------------------------------------------------------------------------------
--------------------
```

[173]:
```
# Reorder the main data frame
df = df.loc[df['ChEMBL ID'].isin(cmEMBL_right_order)].sort_values(by=['ChEMBL␣
 ↪ID'], key=lambda x: x.map(dict(zip(cmEMBL_right_order,␣
 ↪range(len(cmEMBL_right_order))))))

# Resetting index after sorting
df = df.reset_index(drop=True)
```

[174]:
```
df
```

[174]:

	ChEMBL ID	Name
0	CHEMBL2111481	Integrin alpha-4/beta-7
1	CHEMBL2095184	Integrin alpha-4/beta-7
2	CHEMBL3430894	Integrin alpha-IIb/beta-3
3	CHEMBL3430895	Integrin alpha-10/Integrin beta-1
4	CHEMBL3137278	Integrin alpha1/beta1 complex
5	CHEMBL2095226	Integrin alpha-5/beta-1
6	CHEMBL1907599	Integrin alpha-4/beta-1
7	CHEMBL3885597	ITGB1-ITGA9 complex
8	CHEMBL3137268	Integrin alpha2/beta1
9	CHEMBL3883284	Integrin alpha-3/beta-3
10	CHEMBL2093869	Integrin alpha-IIb/beta-3
11	CHEMBL3885596	ITGA6-ITGB1 complex
12	CHEMBL2111461	Integrin alpha-2/beta-3
13	CHEMBL2096675	Integrin alpha-V/beta-5
14	CHEMBL3430893	Integrin alpha-V/alpha-5
15	CHEMBL2111416	Integrin alpha-V/beta-6
16	CHEMBL2111425	Fibronectin receptor (Integrin alpha-5/beta-1 …
17	CHEMBL4106150	Integrin alpha-V/beta-3/alpha-V/beta-5
18	CHEMBL2111362	Integrin alpha-M/beta-2
19	CHEMBL2364172	Integrin alpha-L/beta-2 (LFA-1)
20	CHEMBL2111443	Integrin alpha-V/beta-3 and alpha-IIb/beta 3

21	CHEMBL3430891	Integrin alpha-V/beta-3
22	CHEMBL2111407	Integrin alpha-V/beta-1
23	CHEMBL1907598	Integrin alpha-V/beta-3
24	CHEMBL2096661	Intercellular adhesion molecule (ICAM-1), Inte…
25	CHEMBL4106121	MAC1-CD40L
26	CHEMBL3885595	Integrin alpha-5/Neuronal acetylcholine recept…
27	CHEMBL4523628	Integrin alpha-V/Integrin beta-5
28	CHEMBL3430892	Integrin alpha-V/beta-8
29	CHEMBL4296069	Integrin subunit alpha 2b/beta-3
30	CHEMBL3883325	Integrin alpha-IIb/beta-3
31	CHEMBL4748218	VHL/Polycomb protein EED
32	CHEMBL3137286	EZH2/SUZ12/EED complex
33	CHEMBL3137287	EZH1/SUZ12/EED/AEBP2/RBBP4 complex
34	CHEMBL3301388	EZH2/SUZ12/EED/RBBP7/RBBP4

	UniProt Accessions	Type	\
0	Q00651|P26011	PROTEIN COMPLEX	
1	P13612|P26010	PROTEIN COMPLEX	
2	Q9QUM0|O54890	PROTEIN COMPLEX	
3	P05556|O75578	PROTEIN COMPLEX	
4	P05556|P56199	PROTEIN COMPLEX	
5	P05556|P08648	PROTEIN COMPLEX	
6	P05556|P13612	PROTEIN COMPLEX	
7	P05556|Q13797	PROTEIN COMPLEX	
8	P05556|P17301	PROTEIN COMPLEX	
9	P05106|P26006	PROTEIN COMPLEX	
10	P05106|P08514	PROTEIN COMPLEX	
11	P05556|P23229	PROTEIN COMPLEX	
12	P05106|P17301	PROTEIN COMPLEX	
13	P06756|P18084	PROTEIN COMPLEX	
14	P06756|P08648	PROTEIN COMPLEX	
15	P06756|P18564	PROTEIN COMPLEX	
16	P05106|P05556|P06756|P08648	PROTEIN COMPLEX GROUP	
17	P05106|P06756|P18084	PROTEIN COMPLEX	
18	P05107|P11215	PROTEIN COMPLEX	
19	P20701|P05107	PROTEIN COMPLEX	
20	P05106|P06756|P08514	SELECTIVITY GROUP	
21	O54890|P43406	PROTEIN COMPLEX	
22	P05556|P06756	PROTEIN COMPLEX	
23	P05106|P06756	PROTEIN COMPLEX	
24	P20701|P05362|P05107	PROTEIN COMPLEX	
25	P05107|P11215|P29965	PROTEIN-PROTEIN INTERACTION	
26	P17787|P30926|P32297|P08648	PROTEIN COMPLEX	
27	P43406|O70309	PROTEIN COMPLEX	
28	P06756|P26012	PROTEIN COMPLEX	
29	D3ZAC0|Q8R2H2	PROTEIN COMPLEX	
30	Q9TUN4|Q9TUN3	PROTEIN COMPLEX	

31	O75530|P40337	PROTEIN-PROTEIN INTERACTION
32	Q15910|O75530|Q15022	PROTEIN COMPLEX
33	Q92800|O75530|Q15022|Q6ZN18|Q09028	PROTEIN COMPLEX
34	Q15910|O75530|Q15022|Q09028|Q16576	PROTEIN COMPLEX

	Organism	Compounds	Activities	Tax ID	Species Group Flag
0	Mus musculus	22	55	10090	False
1	Homo sapiens	522	610	9606	False
2	Mus musculus	4	4	10090	False
3	Homo sapiens	1	1	9606	False
4	Homo sapiens	6	6	9606	False
5	Homo sapiens	463	685	9606	False
6	Homo sapiens	1452	2269	9606	False
7	Homo sapiens	10	10	9606	False
8	Homo sapiens	57	62	9606	False
9	Homo sapiens	25	47	9606	False
10	Homo sapiens	2338	3481	9606	False
11	Homo sapiens	1	1	9606	False
12	Homo sapiens	20	21	9606	False
13	Homo sapiens	454	589	9606	False
14	Homo sapiens	6	6	9606	False
15	Homo sapiens	366	509	9606	False
16	Homo sapiens	37	39	9606	False
17	Homo sapiens	8	8	9606	False
18	Homo sapiens	27	30	9606	False
19	Homo sapiens	33	51	9606	False
20	Homo sapiens	49	49	9606	False
21	Mus musculus	4	4	10090	False
22	Homo sapiens	170	222	9606	False
23	Homo sapiens	2001	2705	9606	False
24	Homo sapiens	588	760	9606	False
25	Homo sapiens	6	6	9606	False
26	Homo sapiens	13	25	9606	False
27	Mus musculus	2	2	10090	False
28	Homo sapiens	161	181	9606	False
29	Rattus norvegicus	11	18	10116	False
30	Oryctolagus cuniculus	34	53	9986	False
31	Homo sapiens	2	6	9606	False
32	Homo sapiens	10	16	9606	False
33	Homo sapiens	29	29	9606	False
34	Homo sapiens	7	11	9606	False

Continue the Merge Step First we would like to split *UniProt Accessions* column into 5 columns: *UniProt1, … , Uniprot5*. The reason behind that step is that we discovered that *UniProt Accessions* column contains not just pairs of proteins, but only can contain 3,4 or 5. For rows that contain less than 5 proteins in *UniProt Accessions*, the coressponding *UniProt#* column will contain `None` values.

[175]:
```
df.drop(df.columns.difference(['Name', 'UniProt Accessions',␣
 ↪'Organism','Compounds']), axis=1, inplace=True)

df.head(5)
```

[175]:

	Name	UniProt Accessions	Organism	Compounds
0	Integrin alpha-4/beta-7	Q00651|P26011	Mus musculus	22
1	Integrin alpha-4/beta-7	P13612|P26010	Homo sapiens	522
2	Integrin alpha-IIb/beta-3	Q9QUM0|O54890	Mus musculus	4
3	Integrin alpha-10/Integrin beta-1	P05556|O75578	Homo sapiens	1
4	Integrin alpha1/beta1 complex	P05556|P56199	Homo sapiens	6

[176]:
```
# Split the 'UniProt' column into separate columns based on '|'
uniprot_columns = df['UniProt Accessions'].str.split('|', expand=True)

# Rename the new columns
uniprot_columns.columns = [f'UniProt{i}' for i in range(1, uniprot_columns.
 ↪shape[1] + 1)]
df = df.drop(columns=df.filter(like='UniProt').columns)

# Concatenate the new columns to the original DataFrame
df = pd.concat([df, uniprot_columns], axis=1)

df.head(10)
```

[176]:

	Name	Organism	Compounds	UniProt1	UniProt2	UniProt3	UniProt4	UniProt5
0	Integrin alpha-4/beta-7	Mus musculus	22	Q00651	P26011	None	None	None
1	Integrin alpha-4/beta-7	Homo sapiens	522	P13612	P26010	None	None	None
2	Integrin alpha-IIb/beta-3	Mus musculus	4	Q9QUM0				
3	Integrin alpha-10/Integrin beta-1	Homo sapiens	1	P05556				
4	Integrin alpha1/beta1 complex	Homo sapiens	6	P05556				
5	Integrin alpha-5/beta-1	Homo sapiens	463	P05556				
6	Integrin alpha-4/beta-1	Homo sapiens	1452	P05556				
7	ITGB1-ITGA9 complex	Homo sapiens	10	P05556				
8	Integrin alpha2/beta1	Homo sapiens	57	P05556				
9	Integrin alpha-3/beta-3	Homo sapiens	25	P05106				

```
2   O54890     None     None     None
3   O75578     None     None     None
4   P56199     None     None     None
5   P08648     None     None     None
6   P13612     None     None     None
7   Q13797     None     None     None
8   P17301     None     None     None
9   P26006     None     None     None
```

[177]:
```
df.tail(5)
```

[177]:
```
                                Name               Organism  Compounds  \
30          Integrin alpha-IIb/beta-3  Oryctolagus cuniculus         34
31           VHL/Polycomb protein EED           Homo sapiens          2
32             EZH2/SUZ12/EED complex           Homo sapiens         10
33  EZH1/SUZ12/EED/AEBP2/RBBP4 complex           Homo sapiens         29
34         EZH2/SUZ12/EED/RBBP7/RBBP4           Homo sapiens          7

   UniProt1 UniProt2 UniProt3 UniProt4 UniProt5
30   Q9TUN4   Q9TUN3     None     None     None
31   O75530   P40337     None     None     None
32   Q15910   O75530   Q15022     None     None
33   Q92800   O75530   Q15022   Q6ZN18   Q09028
34   Q15910   O75530   Q15022   Q09028   Q16576
```

After addressing the UniProt columns, we can proceed to merge the main DataFrame with its corresponding SMILES DataFrames obtained from the respective CSV files.

Steps:

- Create a temporary list to store all the DataFrames generated for each row in main_df.
- Iterate through all the rows in main_df.
- Create a temporary DataFrame containing only the row corresponding to the index+1 in main_df. Duplicate this row for each row in the CSV file that corresponds to that row in the *Integrins - common Compounds csv - SMILES only* directory.
- Add a new column to the temporary DataFrame named 'Smiles,' which contains the molecular SMILES values from the CSV file.
- Concatenate all DataFrames in the temporary list into a new main data frame.

[202]:
```
df_before_merging = df

number_of_rows_with_unique_compounds_value = df_before_merging.shape[0]

PRINT(f"The number of rows which correlates to the number of unique compounds␣
 ↪values are -> {number_of_rows_with_unique_compounds_value}")
```

```
--------------------------------------------------------------------------------
--------------------
The number of rows which correlates to the number of unique compounds values are
```

```
-> 35
--------------------------------------------------------------------------------
--------------------
```

[178]:
```
main_df = df
```

[179]:
```
# Path to the directory containing molecules SMILES CSV files.
csv_directory = 'Integrins - common Compounds csv - SMILES only'

# Temo list to store DataFrames after processing
dfs_to_concat = []

for index, row in main_df.iterrows():
    # Get the corresponding CSV file path
    csv_file_path = os.path.join(csv_directory, f'compounds_SMILES_{index+1}.
 ↪csv')
    csv_data = pd.read_csv(csv_file_path)

    # Duplicate the main_df row for each row in the CSV file
    duplicated_rows = pd.DataFrame([row] * len(csv_data), columns=main_df.
 ↪columns)

    # Add a new 'Smiles' column with the values from the CSV file
    duplicated_rows['Smiles'] = csv_data['Smiles'].tolist()

    # Append the processed data frame to the temp list
    dfs_to_concat.append(duplicated_rows)

# Concatenate all data frames in the list into a new data frame
main_df_processed = pd.concat(dfs_to_concat, ignore_index=True)

# Reset the index of the main data frame
main_df_processed = main_df_processed.reset_index(drop=True)

main_df = main_df_processed

PRINT()
print('Done')
PRINT()
```

```
--------------------------------------------------------------------------------
Done
--------------------------------------------------------------------------------
```

[180]:
```
main_df
```

[180]:

	Name	Organism	Compounds	UniProt1	UniProt2	\
0	Integrin alpha-4/beta-7	Mus musculus	22	Q00651	P26011	

```
1         Integrin alpha-4/beta-7  Mus musculus         22   Q00651   P26011
2         Integrin alpha-4/beta-7  Mus musculus         22   Q00651   P26011
3         Integrin alpha-4/beta-7  Mus musculus         22   Q00651   P26011
4         Integrin alpha-4/beta-7  Mus musculus         22   Q00651   P26011
…                              …           …        …       …      …
8934  EZH2/SUZ12/EED/RBBP7/RBBP4  Homo sapiens          7   Q15910   O75530
8935  EZH2/SUZ12/EED/RBBP7/RBBP4  Homo sapiens          7   Q15910   O75530
8936  EZH2/SUZ12/EED/RBBP7/RBBP4  Homo sapiens          7   Q15910   O75530
8937  EZH2/SUZ12/EED/RBBP7/RBBP4  Homo sapiens          7   Q15910   O75530
8938  EZH2/SUZ12/EED/RBBP7/RBBP4  Homo sapiens          7   Q15910   O75530

     UniProt3 UniProt4 UniProt5  \
0        None     None     None
1        None     None     None
2        None     None     None
3        None     None     None
4        None     None     None
…         …       …        …
8934   Q15022   Q09028   Q16576
8935   Q15022   Q09028   Q16576
8936   Q15022   Q09028   Q16576
8937   Q15022   Q09028   Q16576
8938   Q15022   Q09028   Q16576

                                                    Smiles
0     N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…
1     N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
2     N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…
3     N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
4     N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
…                                                   …
8934  CCCc1cc(C)[nH]c(=O)c1CNC(=O)c1cc(-c2ccc(N3CCN(…
8935  Cc1cc(C)c(CNC(=O)c2cc(C3CC3)nc3c2cnn3C(C)C)c(O)n1
8936  Cc1cc(C)c(CNC(=O)c2cc(-c3ccc(N4CCN(C)CC4)nc3)c…
8937  Cc1cc(C)c(CNC(=O)c2cc(-c3ccnc(N4CCN(C)CC4)c3)c…
8938  Cc1cc(C)c(CNC(=O)c2cc(C3CC3)cc3c2cnn3C(C)C)c(=…

[8939 rows x 9 columns]
```

Vefification of the Merge Step After we merged and duplicated our main data frame rows with their coressponding molecule SMILES CSV files, we want to verify that we got all of the data and didnt missed any molecule during the last step.

In order to achive the verification step, we will count for each *Compounds* column value the number of rows and check wether they are equal. e.g. if we got for instance Compounds=22, we will count the number of rows which corresponds to that values, and check if there are indeed 22 such rows

[182]:
```
main_df.to_csv("check_main.csv", index=False)
```

[204]:
```
def is_marge_step_is_validated(df, norwucv) -> bool:
    """
    Function that verifies the merge step by comparing the number of rows for␣
 ↪each compounds values to it's corresponding compounds value.

    Parmeters:

    df: Data frame
    norwucv: Number of rows with unique compounds values (can be duplicated␣
 ↪values)

    Return:

    True if the number of rows for each compounds values is equal to the␣
 ↪corresponding compounds value, else False
    """
    row_counter = 1
    confirmed_values = 0

    for idx, row in df.iterrows():
        # Extract current row compounds value
        compounds_value = row['Compounds']

        # Check whether number of rows that contains current compounds value is␣
 ↪equal to compounds value
        if row_counter == compounds_value:
            confirmed_values+=1
            # Reset row counting
            row_counter=1
            continue

        row_counter+=1

    return norwucv==confirmed_values
```

[205]:
```
merge_check = is_marge_step_is_validated(main_df,␣
 ↪number_of_rows_with_unique_compounds_value)
```

[206]:
```
PRINT(f"True if the merge done successfully, else False --->> {merge_check}")
```

```
--------------------------------------------------------------------------------
--------------------
True if the merge done successfully, else False --->> True
--------------------------------------------------------------------------------
--------------------
```

Remove Compounds Column After we verified that we got the correct number of rows after adding for each row in the main data frame its corresponding extra rows from the compounds CSV file, we can drop *Compounds* column.

[11]:
```
main_df.drop('Compounds', inplace=True, axis=1)
```

[12]:
```
main_df.head(3)
```

[12]:
```
                              Canonical SMILES(RDKit)          Target Pref Name  \
0  N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…  Integrin alpha-4/beta-7
1  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
2  N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…  Integrin alpha-4/beta-7

        Organism UniProt1 UniProt2 UniProt3 UniProt4 UniProt5
0  Mus musculus   Q00651   P26011      NaN      NaN      NaN
1  Mus musculus   Q00651   P26011      NaN      NaN      NaN
2  Mus musculus   Q00651   P26011      NaN      NaN      NaN
```

1.7 Join Step

After we prepared our `main_df` with the correct data for *Integrins* target pref name, we can join the data frame with the data frame from previous project (e.g. `ppi_curaion_search_1033`).

The join will be done by the molecules *SMILES* value (i.e. simplified molecular-input line-entry system)

Load Previous Data Frame

[117]:
```
# Get CSV path
ppi_curation_search_1033_csv_path = r"C:
↪\Users\gavvi\Desktop\Programming\GitHub\DeepLearningResearchStarship\Project␣
↪2 Web Scraping\ppi_curation_search_1033_.csv"
```

[118]:
```
# Load CSV file into data frame
ppi_cs1033_df = pd.read_csv(ppi_curation_search_1033_csv_path)

ppi_cs1033_df.head(5)
```

[118]:
```
  DLiP-ID                              Canonical SMILES(RDKit)       MW  XLogP  \
0  T00000  CCC(C)(C)C(=O)C(=O)N1CCCCC1C(=O)OCCCc1cc(OC)cc…  433.545  3.548
1  T00001  COc1ccccc1C1C2=C(N=c3s/c(=C\c4ccc(/C=C/C(=O)O)…  520.610  5.492
2  T00002     CSc1ccc(-c2c(C#N)c3cccc(Cl)n3c2NCCc2ccccc2)cc1  417.965  7.388
3  T00003  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  519.554  5.147
4  T00004  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  519.554  5.147

   HBA  HBD     PSA  nRotatableBonds  nRings  \
0    6    0   82.14               10       2
1    6    1   80.89                5       6
2    4    1   40.23                6       4
```

```
3    7    2  131.24              10       4
4    7    2  131.24              10       4

                        Target Pref Name Common Target Pref Name    Active
0                           FKBP1A/FK506            FKBP1A/FK506    Active
1  Bcl-2 and Bcl-XL with BAX; BAK and BID       BCL-like/BAX,BAK  Inactive
2                    Neuropilin-1/VEGF-A    Neuropilin-1/VEGF-A    Active
3                              Integrins              Integrins    Active
4                              Integrins              Integrins    Active
```

[ ]:

Change the Smiles Column Name In order to join by the SMILES column, we will change the column name of `main_df` to *Canonical SMILES(RDKit)*

[15]:
```
main_df.rename(columns={'Smiles': 'Canonical SMILES(RDKit)'}, inplace=True)
```

[16]:
```
main_df.head(5)
```

[16]:
```
                              Canonical SMILES(RDKit)         Target Pref Name  \
0  N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…  Integrin alpha-4/beta-7
1  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
2  N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…  Integrin alpha-4/beta-7
3  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
4  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7

       Organism UniProt1 UniProt2 UniProt3 UniProt4 UniProt5
0  Mus musculus   Q00651   P26011      NaN      NaN      NaN
1  Mus musculus   Q00651   P26011      NaN      NaN      NaN
2  Mus musculus   Q00651   P26011      NaN      NaN      NaN
3  Mus musculus   Q00651   P26011      NaN      NaN      NaN
4  Mus musculus   Q00651   P26011      NaN      NaN      NaN
```

Change the *Name* Column Name

After modifying the *S,iles* columm, we need also to change the *Name* column into *Target Pref name* in order to execute the join successfully.

[17]:
```
main_df.rename(columns={'Name': 'Target Pref Name'}, inplace=True)
```

[18]:
```
main_df.head(5)
```

[18]:
```
                              Canonical SMILES(RDKit)         Target Pref Name  \
0  N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…  Integrin alpha-4/beta-7
1  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
2  N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…  Integrin alpha-4/beta-7
3  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
4  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
```

```
        Organism UniProt1 UniProt2 UniProt3 UniProt4 UniProt5
0  Mus musculus   Q00651   P26011      NaN      NaN      NaN
1  Mus musculus   Q00651   P26011      NaN      NaN      NaN
2  Mus musculus   Q00651   P26011      NaN      NaN      NaN
3  Mus musculus   Q00651   P26011      NaN      NaN      NaN
4  Mus musculus   Q00651   P26011      NaN      NaN      NaN
```

Reorder the Columns for Better Visualization

[20]:
```
main_df = main_df[['Canonical SMILES(RDKit)', 'Target Pref Name', 'Organism',␣
↪'UniProt1', 'UniProt2', 'UniProt3', 'UniProt4', 'UniProt5']]
```

[21]:
```
main_df.head(3)
```

[21]:
```
                              Canonical SMILES(RDKit)         Target Pref Name  \
0  N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…  Integrin alpha-4/beta-7
1  N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…  Integrin alpha-4/beta-7
2  N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…  Integrin alpha-4/beta-7

        Organism UniProt1 UniProt2 UniProt3 UniProt4 UniProt5
0  Mus musculus   Q00651   P26011      NaN      NaN      NaN
1  Mus musculus   Q00651   P26011      NaN      NaN      NaN
2  Mus musculus   Q00651   P26011      NaN      NaN      NaN
```

[23]:
```
# Count the number of rows in ppi_cs1033_df containing 'Integrins' before the␣
↪merge
before_merge_count_initial = ppi_cs1033_df[ppi_cs1033_df['Target Pref Name'] ==␣
↪'Integrins'].shape[0]

PRINT(f"Number of rows with 'Integrins' in ppi_cs1033_df initially:␣
↪{before_merge_count_initial}")
PRINT(f"Number of rows in main_df: {main_df.shape[0]}")
```

```
--------------------------------------------------------------------------------
--------------------
Number of rows with 'Integrins' in ppi_cs1033_df initially: 8684
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Number of rows in main_df: 8939
--------------------------------------------------------------------------------
--------------------
```

Join Between Two Data Frames

[24]:
```
join_columns = ['Target Pref Name', 'Canonical SMILES(RDKit)']
```

[25]:
```
partial_main_df = main_df[join_columns]
#partial_ppi_cs1033_df = ppi_cs1033_df[join_columns]
```

[26]:
```
partial_main_df.head(5)
```

[26]:

	Target Pref Name	Canonical SMILES(RDKit)
0	Integrin alpha-4/beta-7	N=C(N)NCCC[C@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[…
1	Integrin alpha-4/beta-7	N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
2	Integrin alpha-4/beta-7	N#Cc1ccc(-c2ccc(C[C@H](NC(=O)[C@H](CCCNC(=N)N)…
3	Integrin alpha-4/beta-7	N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…
4	Integrin alpha-4/beta-7	N=C(N)NCCC[C@H](NC(=O)CCCC[C@@H]1SC[C@@H]2NC(=…

[27]:
```
ppi_cs1033_df.head(5)
```

[27]:

	DLiP-ID	Canonical SMILES(RDKit)	MW	XLogP	HBA	HBD	PSA	nRotatableBonds	nRings	Target Pref Name	Common Target Pref Name	Active
0	T00000	CCC(C)(C)C(=O)C(=O)N1CCCCC1C(=O)OCCCc1cc(OC)cc…	433.545	3.548	6	0	82.14	10	2	FKBP1A/FK506	FKBP1A/FK506	Active
1	T00001	COc1ccccc1C1C2=C(N=c3s/c(=C\c4ccc(/C=C/C(=O)O)…	520.610	5.492	6	1	80.89	5	6	Bcl-2 and Bcl-XL with BAX; BAK and BID	BCL-like/BAX,BAK	Inactive
2	T00002	CSc1ccc(-c2c(C#N)c3cccc(Cl)n3c2NCCc2ccccc2)cc1	417.965	7.388	4	1	40.23	6	4	Neuropilin-1/VEGF-A	Neuropilin-1/VEGF-A	Active
3	T00003	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554	5.147	7	2	131.24	10	4	Integrins	Integrins	Active
4	T00004	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554	5.147	7	2	131.24	10	4	Integrins	Integrins	Active

[28]:
```
partial_ppi_cs1033_df_ = ppi_cs1033_df
partial_main_df_ = partial_main_df
```

[29]:
```
partial_ppi_cs1033_df_
```

[29]:

	DLiP-ID	Canonical SMILES(RDKit)	MW
0	T00000	CCC(C)(C)C(=O)C(=O)N1CCCCC1C(=O)OCCCc1cc(OC)cc…	433.545
1	T00001	COc1ccccc1C1C2=C(N=c3s/c(=C\c4ccc(/C=C/C(=O)O)…	520.610
2	T00002	CSc1ccc(-c2c(C#N)c3cccc(Cl)n3c2NCCc2ccccc2)cc1	417.965
3	T00003	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554
4	T00004	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554
…	…	…	…
25812	T00010	O=C(NC1CCCCCC1)C(Cc1ccccc1)NS(=O)(=O)c1cccc2ns…	458.609

```
25813  T00011  CC(C)c1ccccc1Sc1ccc(-c2cc(N3CCC(C(=O)O)CC3)ncn…  501.574
25814  T00012  C/C(=N\Nc1nc2c(F)cccc2s1)c1ccc(-c2ccc(Cl)c(C(=…  429.860
25815  T00013  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…  502.593
25816  T00014  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…  502.593

       XLogP  HBA  HBD     PSA  nRotatableBonds  nRings  \
0      3.548    6    0   82.14               10       2
1      5.492    6    1   80.89                5       6
2      7.388    4    1   40.23                6       4
3      5.147    7    2  131.24               10       4
4      5.147    7    2  131.24               10       4
…      …  …  …     …              …      …
25812  4.598    6    2  101.05                7       4
25813  7.045    5    1   66.32                6       4
25814  5.482    6    2   87.72                5       4
25815  2.100    5    4  144.91                9       3
25816  2.100    5    4  144.91                9       3

                            Target Pref Name Common Target Pref Name  \
0                                FKBP1A/FKB506            FKBP1A/FKB506
1      Bcl-2 and Bcl-XL with BAX; BAK and BID        BCL-like/BAX,BAK
2                         Neuropilin-1/VEGF-A     Neuropilin-1/VEGF-A
3                                   Integrins               Integrins
4                                   Integrins               Integrins
…                                       …                       …
25812                               Integrins               Integrins
25813                               Integrins               Integrins
25814  Bcl-2 and Bcl-XL with BAX; BAK and BID        BCL-like/BAX,BAK
25815                               Integrins               Integrins
25816                               Integrins               Integrins

         Active
0        Active
1      Inactive
2        Active
3        Active
4        Active
…         …
25812  Inactive
25813    Active
25814    Active
25815    Active
25816  Inactive

[25817 rows x 12 columns]
```

Our tactic:

We are going to take only the two columns '*Target Pref Name*', '*Canonical SMILES(RDKit)*' from the two data frames and join on the *SMILES* values.

The result in going to be saved in new data frame called "*result_df*' that going to have three columns, which are:

- Target Pref Name_ppi: Column which consists only of Integrins values (because thats the target pref. name we are joinning on)
- Canonical SMILES(RDKit): The SMILES values of the molecules we found match on
- Target Pref Name_main: Column with more detailed target pref. name value; our desired value that we want to extract

Next step is switching between the values of "*Target Pref Name*' column in *ppi_cs1033_df* data frame

[30]:
```
# Check the number of rows in partial_ppi_cs1033_df_ containing 'Integrins'␣
 ↪before the merge
before_merge_count_initial =␣
 ↪partial_ppi_cs1033_df_[partial_ppi_cs1033_df_['Target Pref Name'] ==␣
 ↪'Integrins'].shape[0]
print("Number of rows with 'Integrins' in partial_ppi_cs1033_df_ initially:",␣
 ↪before_merge_count_initial)

# Select rows in partial_ppi_cs1033_df_ where 'Target Pref Name' is 'Integrins'
integrins_rows = partial_ppi_cs1033_df_[partial_ppi_cs1033_df_['Target Pref␣
 ↪Name'] == 'Integrins'].copy()

# Merge the selected rows from partial_ppi_cs1033_df_ with partial_main_df_
result_df = pd.merge(integrins_rows[['Target Pref Name', 'Canonical␣
 ↪SMILES(RDKit)']],
                     partial_main_df_[['Canonical SMILES(RDKit)', 'Target Pref␣
 ↪Name']],
                     how='left',
                     left_on='Canonical SMILES(RDKit)',
                     right_on='Canonical SMILES(RDKit)',
                     suffixes=('_ppi', '_main'))

# Check for missing values in the merged data frame
missing_values = result_df[result_df['Target Pref Name_main'].isnull()]
```

```
Number of rows with 'Integrins' in partial_ppi_cs1033_df_ initially: 8684
```

Understanding the Results The next step include understanding our results. Which means: - Check if there are molecules that we didn't find any match between the SMILES in DLiP and ChEMB - Check how many rows, i.e. matches, we got in our result data frame

[34]:
```
missing_values
```

[34]:
```
      Target Pref Name_ppi                                Canonical SMILES(RDKit)  \
76                Integrins  CC(=O)CN[C@H]1CSSC(C)(C)[C@H](C(N)=O)NC(=O)[C@…
77                Integrins  CC(=O)CN[C@H]1CSSC(C)(C)[C@H](C(N)=O)NC(=O)[C@…
225               Integrins  C[C@@]1(C(=O)N[C@@H](Cc2ccc(NC(=O)c3c(Cl)cncc3…
257               Integrins  O=C([O-])C(F)(F)F.[NH3+]Cc1ccc(NC(=O)N2C(=O)CC…
258               Integrins  O=C([O-])C(F)(F)F.[NH3+]Cc1ccc(NC(=O)N2C(=O)CC…
…                     …                                                  …
13979             Integrins  COC(=O)c1c(C(C)C)cc(O)c2c(O)c3c(c(O)c12)C(=O)c…
14053             Integrins         O=C(O)CCN1Cc2ccc(NC(=O)CCCC3CCNCC3)cc2C1=O
14063             Integrins  COc1cc(CC(=O)N2C[C@@H](F)C[C@H]2COc2ccc(C(=O)O…
14188             Integrins  COc1ccccc1NC(=O)[C@@H]1OCO[C@H]1C(=O)N[C@@H](C…
14208             Integrins      CC(CC(=O)O)N1Cc2ccc(NC(=O)CCCC3CCNCC3)cc2C1=O

      Target Pref Name_main
76                      NaN
77                      NaN
225                     NaN
257                     NaN
258                     NaN
…                       …
13979                   NaN
14053                   NaN
14063                   NaN
14188                   NaN
14208                   NaN

[444 rows x 3 columns]
```

[33]:
```
PRINT(f"The number of rows where we didn't find any match between the SMILES in␣
 ↪DLiP and ChEMBL is -> {missing_values.shape[0]}")
```

```
--------------------------------------------------------------------------------
--------------------
The number of rows where we didn't find any match between the SMILES in DLiP and
ChEMBL is -> 444
--------------------------------------------------------------------------------
--------------------
```

As we can see. there are 444 rows in *ppi_cs1033_df* that dont have match with *main_df* by the SMILES values. Which means there are 444 moleculs in *ppi_cs1033_df* that we cant extract their extended *target pref name* from *main_df'* data frame that we generated from *chEMBL* data base.

Therefore, we will leave those rows as they was before, and modify one the rows we found match with between both data frames.

[35]:
```
result_df
```

[35]:
```
      Target Pref Name_ppi                                Canonical SMILES(RDKit)  \
0                 Integrins  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
```

```
1                Integrins  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
2                Integrins  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
3                Integrins  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
4                Integrins  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
…                      …                                                 …
14283            Integrins  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14284            Integrins  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14285            Integrins  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14286            Integrins  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14287            Integrins  Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…

                Target Pref Name_main
0            Integrin alpha-4/beta-7
1            Integrin alpha-4/beta-1
2            Integrin alpha-4/beta-7
3            Integrin alpha-4/beta-1
4            Integrin alpha-4/beta-7
…                              …
14283          Integrin alpha2/beta1
14284  Integrin alpha1/beta1 complex
14285        Integrin alpha-5/beta-1
14286        Integrin alpha-4/beta-1
14287          Integrin alpha2/beta1

[14288 rows x 3 columns]
```

[36]:
```
PRINT(f"Number of rows, i.e. matches, found between the two data frames is ->␣
  ↪{result_df.shape[0]}")
```

```
--------------------------------------------------------------------------------
--------------------
Number of rows, i.e. matches, found between the two data frames is -> 14288
--------------------------------------------------------------------------------
--------------------
```

As observed earlier, certain molecules have multiple matches in the *SMILES* values within the second data frame. This has led to an increase in both data volume and the number of rows after the merging process. Moreover, we have encountered some `Null` values in certain rows, indicating that there was no match for some molecules in the *ppi_cs1033_df* regarding the *SMILES* values with *main_df*. Consequently, we obtained no match for the target preference name value associated with the Integrins target.

For visualization, let's examine the number of rows with `Integrins` values before and after the merge. Furthermore, we will visualize the count of rows for which no match was found, as described above.

[140]:
```

```

```
PRINT(f'Before merging, we had -> {before_merge_count_initial} "Integrins"␣
 ↪values in ppi_cs1033_df, and after merging we got -> {result_df.shape[0]}␣
 ↪values.\nThat indicates that we added {result_df.shape[0] -␣
 ↪before_merge_count_initial} "Target Pref. Name" values')

PRINT(f'The number of unmatched values of SMILES in partial_main_df to␣
 ↪partial_ppi_cs1033_df are -> {missing_values.shape[0]}')
```

```
--------------------------------------------------------------------------------
--------------------
Before merging, we had -> 8684 "Integrins" values in ppi_cs1033_df, and after
merging we got -> 14288 values.
That indicates that we added 5604 "Target Pref. Name" values
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
The number of unmatched values of SMILES in partial_main_df to
partial_ppi_cs1033_df are -> 444
--------------------------------------------------------------------------------
--------------------
```

Saving the Data Frames as CSV

[425]:
```
result_df.to_csv("res_df.csv", index=False)
```

[147]:
```
missing_values.to_csv("unmatched_values.csv", index=False)
```

[150]:
```
main_df.to_csv("chEMBL_Integrins.csv", index=False)
```

[91]:
```
result_df = pd.read_csv("res_df.csv")



result_df
```

[91]:

	Target Pref Name_ppi	Canonical SMILES(RDKit) \
0	Integrins	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
1	Integrins	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
2	Integrins	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
3	Integrins	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
4	Integrins	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…
…	…	…
14283	Integrins	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14284	Integrins	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14285	Integrins	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14286	Integrins	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…
14287	Integrins	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…

```
                Target Pref Name_main
0              Integrin alpha-4/beta-7
1              Integrin alpha-4/beta-1
2              Integrin alpha-4/beta-7
3              Integrin alpha-4/beta-1
4              Integrin alpha-4/beta-7
…                                  …
14283            Integrin alpha2/beta1
14284  Integrin alpha1/beta1 complex
14285          Integrin alpha-5/beta-1
14286          Integrin alpha-4/beta-1
14287            Integrin alpha2/beta1

[14288 rows x 3 columns]
```

1.8 Switch Between Old and New Target Pref. Name Values

[92]:
```
ppi_cs1033_df.head(5)
```

[92]:
```
  DLiP-ID                              Canonical SMILES(RDKit)       MW  XLogP  \
0  T00000  CCC(C)(C)C(=O)C(=O)N1CCCCC1C(=O)OCCCc1cc(OC)cc…  433.545  3.548
1  T00001  COc1ccccc1C1C2=C(N=c3s/c(=C\c4ccc(/C=C/C(=O)O)…  520.610  5.492
2  T00002     CSc1ccc(-c2c(C#N)c3cccc(Cl)n3c2NCCc2ccccc2)cc1  417.965  7.388
3  T00003  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  519.554  5.147
4  T00004  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  519.554  5.147

   HBA  HBD     PSA  nRotatableBonds  nRings  \
0    6    0   82.14               10       2
1    6    1   80.89                5       6
2    4    1   40.23                6       4
3    7    2  131.24               10       4
4    7    2  131.24               10       4

                         Target Pref Name Common Target Pref Name    Active
0                            FKBP1A/FKK506            FKBP1A/FKK506    Active
1  Bcl-2 and Bcl-XL with BAX; BAK and BID       BCL-like/BAX,BAK  Inactive
2                     Neuropilin-1/VEGF-A    Neuropilin-1/VEGF-A    Active
3                                Integrins              Integrins    Active
4                                Integrins              Integrins    Active
```

[93]:
```
result_df.drop('Target Pref Name_ppi', inplace=True, axis=1)

result_df.head(3)
```

[93]:
```
                              Canonical SMILES(RDKit)    Target Pref Name_main
0  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  Integrin alpha-4/beta-7
1  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  Integrin alpha-4/beta-1
```

```
2  COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…  Integrin alpha-4/beta-7
```

[94]:
```
print(ppi_cs1033_df.columns)
print(result_df.columns)
```

```
Index(['DLiP-ID', 'Canonical SMILES(RDKit)', 'MW', 'XLogP', 'HBA', 'HBD',
       'PSA', 'nRotatableBonds', 'nRings', 'Target Pref Name',
       'Common Target Pref Name', 'Active'],
      dtype='object')
Index(['Canonical SMILES(RDKit)', 'Target Pref Name_main'], dtype='object')
```

[132]:
```
# Create an empty list to store the modified rows

modified_rows = []

# Variable in order ro verify we indeed visited in each row
count = 0
count_unmatched_rows = 0
# Loop through each row in ppi_cs1033_df
for index, row in ppi_cs1033_df.iterrows():
    target_pref_name = row['Target Pref Name']

    # Check if the 'Target Pref Name' is 'Integrins'
    if target_pref_name == 'Integrins':
        # Find matches in result_df based on 'Canonical SMILES(RDKit)'
        matches = result_df[result_df['Canonical SMILES(RDKit)'] ==␣
 ↪row['Canonical SMILES(RDKit)']]['Target Pref Name_main'].tolist()
        #print(matches)
        count+=1

        if pd.isna(matches[0]):
            modified_rows.append(row)
            count_unmatched_rows+=1
            continue
        # Duplicate the row for each match and update 'Target Pref Name'
        for match_value in matches:
            duplicated_row = row.copy()
            duplicated_row['Target Pref Name'] = match_value
            modified_rows.append(duplicated_row)
    else:
        count+=1
        # If 'Target Pref Name' is not 'Integrins', keep the original row
        modified_rows.append(row)


# Create a new DataFrame with the modified rows
modified_df = pd.DataFrame(modified_rows)
```

```
# Reset the index to bring 'Canonical SMILES(RDKit)' back as a column
modified_df.reset_index(drop=True, inplace=True)

PRINT(f'Done.\nVisited in :{count} rows, which means skipped over␣
 ↪{ppi_cs1033_df.shape[0]-count} rows (should be 0)')
PRINT(f'Number of unmatched rows we got is -> {count_unmatched_rows}, and we␣
 ↪know we should get 444')
```

```
--------------------------------------------------------------------------------
--------------------
Done.
Visited in :25817 rows, which means skipped over 0 rows (should be 0)
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
Number of unmatched rows we got is -> 444, and we know we should get 444
--------------------------------------------------------------------------------
--------------------
```

[137]:
```
modified_df
```

[137]:

	DLiP-ID	Canonical SMILES(RDKit)	MW	\
0	T00000	CCC(C)(C)C(=O)C(=O)N1CCCCC1C(=O)OCCCc1cc(OC)cc…	433.545	
1	T00001	COc1ccccc1C1C2=C(N=c3s/c(=C\c4ccc(/C=C/C(=O)O)…	520.610	
2	T00002	CSc1ccc(-c2c(C#N)c3cccc(Cl)n3c2NCCc2ccccc2)cc1	417.965	
3	T00003	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554	
4	T00003	COc1cccc(OC)c1-c1ccc(C[C@H](NC(=O)[C@@H]2CCCN2…	519.554	
…	…	…	…	
65560	T00014	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…	502.593	
65561	T00014	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…	502.593	
65562	T00014	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…	502.593	
65563	T00014	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…	502.593	
65564	T00014	Cc1cc(C)cc(S(=O)(=O)N2CCC[C@H]2C(=O)N[C@@H](CN…	502.593	

	XLogP	HBA	HBD	PSA	nRotatableBonds	nRings	\
0	3.548	6	0	82.14	10	2	
1	5.492	6	1	80.89	5	6	
2	7.388	4	1	40.23	6	4	
3	5.147	7	2	131.24	10	4	
4	5.147	7	2	131.24	10	4	
…	…	…	…	…	…	…	
65560	2.100	5	4	144.91	9	3	
65561	2.100	5	4	144.91	9	3	
65562	2.100	5	4	144.91	9	3	
65563	2.100	5	4	144.91	9	3	

```
65564  2.100    5    4  144.91                9       3

                            Target Pref Name Common Target Pref Name  \
0                                  FKBP1A/FK506            FKBP1A/FK506
1      Bcl-2 and Bcl-XL with BAX; BAK and BID       BCL-like/BAX,BAK
2                        Neuropilin-1/VEGF-A    Neuropilin-1/VEGF-A
3                     Integrin alpha-4/beta-7              Integrins
4                     Integrin alpha-4/beta-1              Integrins
…                                        …                    …
65560                   Integrin alpha2/beta1              Integrins
65561           Integrin alpha1/beta1 complex              Integrins
65562                 Integrin alpha-5/beta-1              Integrins
65563                 Integrin alpha-4/beta-1              Integrins
65564                   Integrin alpha2/beta1              Integrins

         Active
0        Active
1      Inactive
2        Active
3        Active
4        Active
…         …
65560  Inactive
65561  Inactive
65562  Inactive
65563  Inactive
65564  Inactive

[65565 rows x 12 columns]
```

[146]:
```
PRINT(f'The number of rows after the switch step -> {modified_df.
 ↪shape[0]}\nNumber of added rows to ppi_cs1033_df -> {modified_df.shape[0] -␣
 ↪ppi_cs1033_df.shape[0]}')
PRINT(f'The number of rows before modifying was {ppi_cs1033_df.shape[0]}, that␣
 ↪should match to the diffrence between the modified data drame and the␣
 ↪original\nCheck ----> {modified_df.shape[0]-(modified_df.shape[0] -␣
 ↪ppi_cs1033_df.shape[0])}')

counted_non_modified_rows = (modified_df['Target Pref Name']=='Integrins').sum()
PRINT(f'The number of unmodified rows expected to be -> {missing_values.
 ↪shape[0]}, and we got -> {counted_non_modified_rows}')
```

```
--------------------------------------------------------------------------------
--------------------
The number of rows after the switch step -> 65565
Number of added rows to ppi_cs1033_df -> 39748
--------------------------------------------------------------------------------
--------------------
```

```
--------------------------------------------------------------------------------
--------------------
The number of rows before modifying was 25817, that should match to the
diffrence between the modified data drame and the original
Check ----> 25817
--------------------------------------------------------------------------------
--------------------
--------------------------------------------------------------------------------
--------------------
The number of unmodified rows expected to be -> 444, and we got -> 444
--------------------------------------------------------------------------------
--------------------
```

1.9 Save the Resulted Data Frames as CSV Files

After me verified that we got the expected data frames, we can save them as CSV files

[143]:
```
modified_df.to_csv('ppi_cs1033_extended.csv', index=False)

PRINT('Saved')
```

```
--------------------------------------------------------------------------------
--------------------
Saved
--------------------------------------------------------------------------------
--------------------
```

[ ]:
```

```