IIA PROJECT
DEADLINE-3
BIO. SYNC

Team members: Aniket Panchal 2021448, Siddharth Anand 2021494, Tanishq Tiwari 2021496

We have to achieve the following tasks.

• Schema matching and mapping
• ETL/Data exchange/propagation
• Adding a new data source or removing a data source

For Schema matching and mapping for pre-existing data sources we had, we already presented them in the previous milestone. Below are the mentions from the previous Milestone

This is the ENTITY MAPPING

```
global_mapping = {

        'atc_classifications': ['atc_classifications', None, None],
        'availability_type': ['availability_type', None, None],
        'biotherapeutic': ['biotherapeutic', None, None],
        'black_box_warning': ['black_box_warning', None, None],
        'chebi_par_id': ['chebi_par_id', None, None],
        'chemical_probe': ['chemical_probe', None, None],
        'chirality': ['chirality', None, None],
        'cross_references': ['cross_references', None, None],
        'dosed_ingredient': ['dosed_ingredient', None, None],
        'first_approval': ['first_approval', None, None],
        'first_in_class': ['first_in_class', None, None],
        'helm_notation': ['helm_notation', None, None],
        'indication_class': ['indication_class', None, None],
        'inorganic_flag': ['inorganic_flag', None, None],
        'max_phase': ['max_phase', None, None],
        'molecule_chembl_id': ['molecule_chembl_id', None, None],
        'molecule_hierarchy': ['molecule_hierarchy', None, None],
        # 'molecule_properties': ['molecule_properties', None, None],
        # 'molecule_structures': ['molecule_structures', None, None],
        'molecule_synonyms': ['molecule_synonyms', None, None],
        'molecule_type': ['molecule_type', None, None],
        'natural_product': ['natural_product', None, None],
        'oral': ['oral', None, None],
        'parenteral': ['parenteral', None, None],
        'polymer_flag': ['polymer_flag', None, None],
        'prodrug': ['prodrug', None, None],
        'structure_type': ['structure_type', None, None],
        'therapeutic_flag': ['therapeutic_flag', None, None],
        'topical': ['topical', None, None],
        'usan_stem': ['usan_stem', None, None],
        'usan_stem_definition': ['usan_stem_definition', None, None],
        'usan_substem': ['usan_substem', None, None],
        'usan_year': ['usan_year', None, None],
        'withdrawn_flag': ['withdrawn_flag', None, None],


        #MOLECULAR PROPERTIES
        'syn_type': ['syn_type', None, None],
        'alogp': ['alogp', None, None],
        'aromatic_rings': ['aromatic_rings', None, None],
        'cx_logd': ['cx_logd', None, None],
        'cx_logp': ['cx_logp', None, None],
        'cx_most_pka1': ['cx_most_pka1', None, None],
        'cx_most_pka2': ['cx_most_pka2', None, None],
        'full_molformula': ['full_molformula', 'formula', None],
        'full_mwt': ['full_mwt', 'molecular_weight', None],
        # 'hba': ['hba', 'h_bond_acceptor_count', None],
        'hba_lipinski': ['hba_lipinski', None, None],
        # 'hbd': ['hbd', 'h_bond_donor_count', None],
        'hbd_lipinski': ['hbd_lipinski', None, None],
        'heavy_atoms': ['heavy_atoms', 'heavy_atom_count', None],
        'molecular_species': ['molecular_species', None, None],
        'mw_freebase': ['mw_freebase', None, None],
        'mw_monoisotopic': ['mw_monoisotopic', 'monoisotopic_mass', None],
        'np_likeness_score': ['np_likeness_score', None, None],
        'num_lipinski_ro5_violations': ['num_lipinski_ro5_violations',
None, None],
        'num_ro5_violations': ['num_ro5_violations', None, None],
        'psa': ['psa', None, None],
        'qed_weighted': ['qed_weighted', None, None],
        'ro3_pass': ['ro3_pass', None, None],
        'rtb': ['rtb', None, None],


        #MOLECULAR STRUCTURE
        'canonical_smiles': ['canonical_smiles', 'canonical_smiles',
None],
        'isomeric_smiles': [None, 'isomeric_smiles', None],
        'molfile': ['molfile', None, None],
        'standard_inchi': ['standard_inchi', 'inchi', None],
        'standard_inchi_key': ['standard_inchi_key', 'inchikey', None],



        'atom_stereo_count': [None, 'atom_stereo_count', None],
        'bond_stereo_count': [None, 'bond_stereo_count', None],
        'cactvs_fingerprint': [None, 'cactvs_fingerprint', None],
        # 'canonical_smiles': [None, 'canonical_smiles', None],
        'charge': [None, 'charge', None],
        'cid': [None, 'cid', None],
        'complexity': [None, 'complexity', None],
        'conformer_id_3d': [None, 'conformer_id_3d', None],
        'conformer_rmsd_3d': [None, 'conformer_rmsd_3d', None],
        'coordinate_type': [None, 'coordinate_type', None],
        'covalent_unit_count': [None, 'covalent_unit_count', None],
        'defined_atom_stereo_count': [None, 'defined_atom_stereo_count',
None],
        'defined_bond_stereo_count': [None, 'defined_bond_stereo_count',
None],
        'effective_rotor_count_3d': [None, 'effective_rotor_count_3d',
None],
        'elements': [None, 'elements', None],
        'exact_mass': [None, 'exact_mass', None],
        'feature_selfoverlap_3d': [None, 'feature_selfoverlap_3d', None],
        'fingerprint': [None, 'fingerprint', None],
        'h_bond_acceptor_count': [None, 'h_bond_acceptor_count', None],
        'h_bond_donor_count': [None, 'h_bond_donor_count', None],
        'heavy_atom_count': [None, 'heavy_atom_count', None],
        # 'inchi': [None, 'inchi', None],
        # 'inchikey': [None, 'inchikey', None],
        # 'isomeric_smiles': [None, 'isomeric_smiles', None],
        'isotope_atom_count': [None, 'isotope_atom_count', None],
        'iupac_name': [None, 'iupac_name', None],
        'mmff94_energy_3d': [None, 'mmff94_energy_3d', None],
        'mmff94_partial_charges_3d': [None, 'mmff94_partial_charges_3d',
None],
        # 'molecular_formula': [None, 'molecular_formula', None],
        # 'molecular_weight': [None, 'molecular_weight', None],
        'monoisotopic_mass': [None, 'monoisotopic_mass', None],
        'multipoles_3d': [None, 'multipoles_3d', None],
        'pharmacophore_features_3d': [None, 'pharmacophore_features_3d',
None],
        'rotatable_bond_count': [None, 'rotatable_bond_count', None],
        'shape_fingerprint_3d': [None, 'shape_fingerprint_3d', None],
        'shape_selfoverlap_3d': [None, 'shape_selfoverlap_3d', None],
        'tpsa': [None, 'tpsa', None],
        'undefined_atom_stereo_count': [None,
'undefined_atom_stereo_count', None],
        'undefined_bond_stereo_count': [None,
'undefined_bond_stereo_count', None],
        'volume_3d': [None, 'volume_3d', None],
        'xlogp': [None, 'xlogp', None],

    }
```

As we have used the virtualised approach, we couldn't find any specific use of ETL tools.
Now,
On adding new data source UNICHEM: A database storing drug/ compound details and information
We created a new global mapping.
As in our approach, we have well-maintained global in use of biochemical data sources, like PubChem, CHEMBL, and UNICHEM, where there is consistency regarding the data accuracy and
For example: taking the column "INCHI-KEY" into consideration, the values present in the CHEMBL data source are the same as those of "inchi-key" present in the UNICHEM data source.
Thus, the probability of entries mismatch is low.
We did not need to use the entity matching algorithms or any other to maintain the incremental view.

Because when a new data source is added or removed, we are not storing that data on our local systems. Despite that, we are using our global definition to align the incoming data according to our needs, and if that data source is not present there, the Main query won't be federated to that data source, and we won't be doing the Natural join for the columns of that data source, thus it removes the possibility of any discrepancy getting created based on the addition and deletion of any table entry or data source.

In case the data is removed from the data source, According to our global definition, instead of showing any error, It will present a null object stating no information regarding that field.

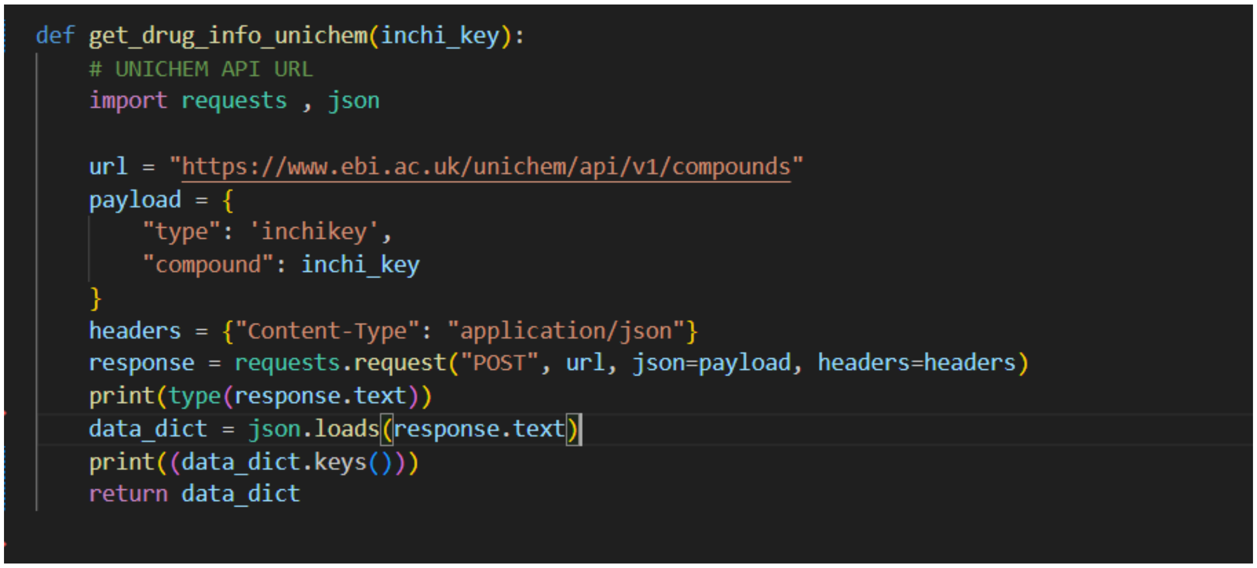
For example:

If we have 3 data sources:	DataA(a,b,c,d)
	DataB(a,c,e)
	DataC(a,c,d)

And we remove the DataA from our application.
Instead of federating and getting errors or any discrepancy for that data source. We won't be making a call for that data source API, and we get that data from the other two data sources having (c,d,e) and for attribute b, we will be presenting a null value,
Moreover, as we are not storing the incoming data from these sources, we won't face any challenges regarding the null values.

UNICHEM DATA SOURCE:

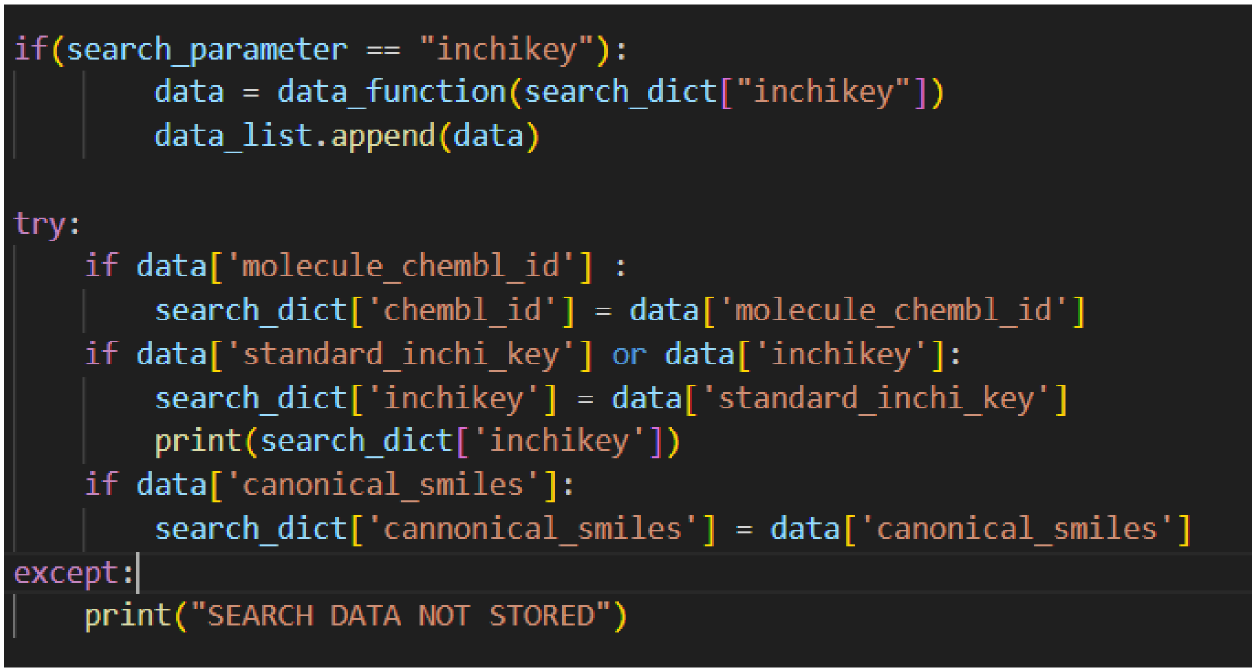
This is a new data source UNICHEM, whose API is shown below.



```python
def get_drug_info_unichem(inchi_key):
    # UNICHEM API URL
    import requests , json

    url = "https://www.ebi.ac.uk/unichem/api/v1/compounds"
    payload = {
        "type": 'inchikey',
        "compound": inchi_key
    }
    headers = {"Content-Type": "application/json"}
    response = requests.request("POST", url, json=payload, headers=headers)
    print(type(response.text))
    data_dict = json.loads(response.text)
    print((data_dict.keys()))
    return data_dict
```

This data source takes “inchi key” as a search parameter instead of drug_name.
So when we enter a drug name, the inchi key will be derived from the CHEMBL database corresponding to that drug name, and then the inchi key will be passed to the UNICHEM API.

In this ‘search_dict’, we will store the inchi key and smiles so that if any other data source takes some another search parameter, it will directly take it from this dictionary.



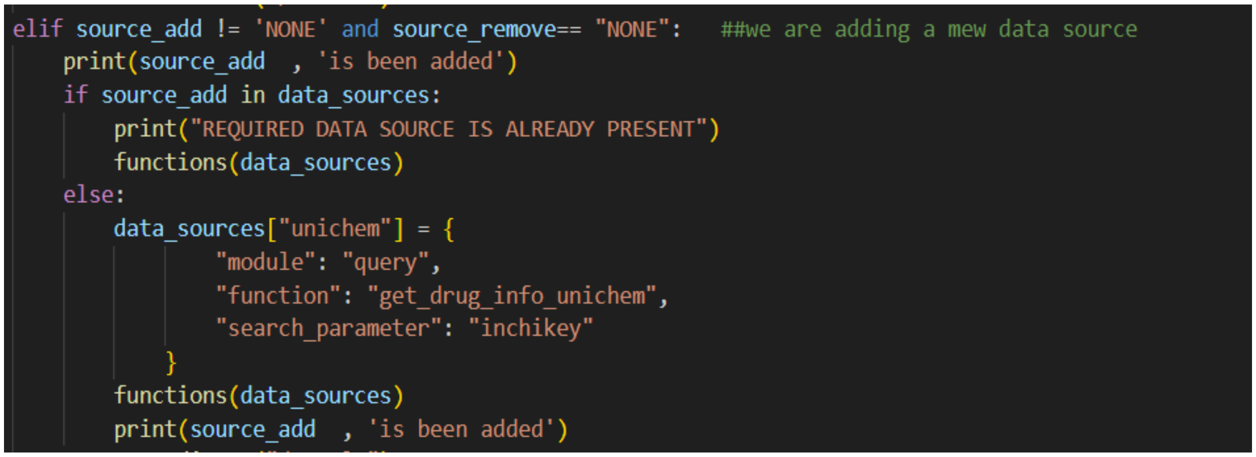
```python
if(search_parameter == "inchikey"):
        data = data_function(search_dict["inchikey"])
        data_list.append(data)

try:
    if data['molecule_chembl_id'] :
        search_dict['chembl_id'] = data['molecule_chembl_id']
    if data['standard_inchi_key'] or data['inchikey']:
        search_dict['inchikey'] = data['standard_inchi_key']
        print(search_dict['inchikey'])
    if data['canonical_smiles']:
        search_dict['cannonical_smiles'] = data['canonical_smiles']
except:
    print("SEARCH DATA NOT STORED")
```

ADDING UNICHEM DATA SOURCE

We will take user input on whether to add or remove the data source, if the data source is already present, than we will do nothing; otherwise, we will use the UNICHEM data source and process accordingly.
On adding the data source, we will manually change the global mapping for the reasons described above.

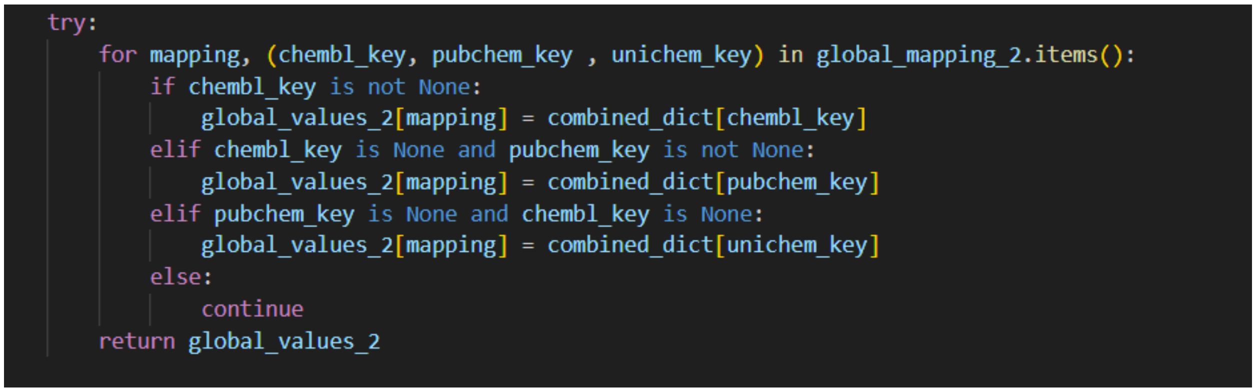
This new data source UNICHEM which has the below attributes
So the new global mapping will be the same as the previous mapping, but it will have these four extra attributes also.

```
        'compounds':[None, None , 'compounds'],
        'notFound':[None, None , 'notFound'],
        'response':[None, None , 'response'],
        'totalCompounds':[None, None , 'totalCompounds'],
```



```python
elif source_add != 'NONE' and source_remove== "NONE":   ##we are adding a mew data source
    print(source_add  , 'is been added')
    if source_add in data_sources:
        print("REQUIRED DATA SOURCE IS ALREADY PRESENT")
        functions(data_sources)
    else:
        data_sources["unichem"] = {
                "module": "query",
                "function": "get_drug_info_unichem",
                "search_parameter": "inchikey"
            }
        functions(data_sources)
        print(source_add  , 'is been added')
```

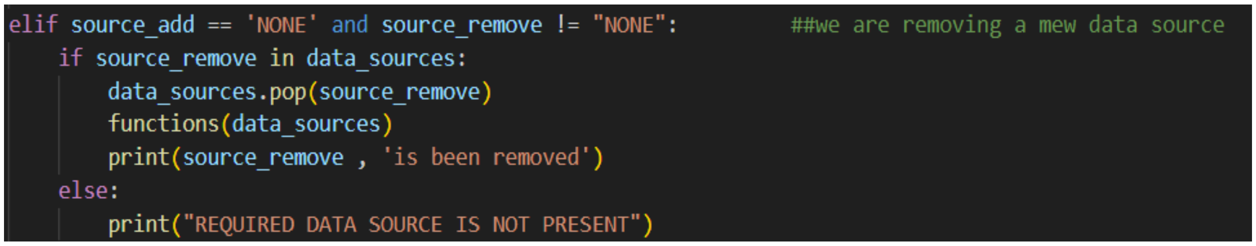
This is how we will make the global schema for storing the data from all the data sources, that is, CHEMBL, PubChem and UNICHEM.



```python
    try:
        for mapping, (chembl_key, pubchem_key , unichem_key) in global_mapping_2.items():
            if chembl_key is not None:
                global_values_2[mapping] = combined_dict[chembl_key]
            elif chembl_key is None and pubchem_key is not None:
                global_values_2[mapping] = combined_dict[pubchem_key]
            elif pubchem_key is None and chembl_key is None:
                global_values_2[mapping] = combined_dict[unichem_key]
            else:
                continue
        return global_values_2
```

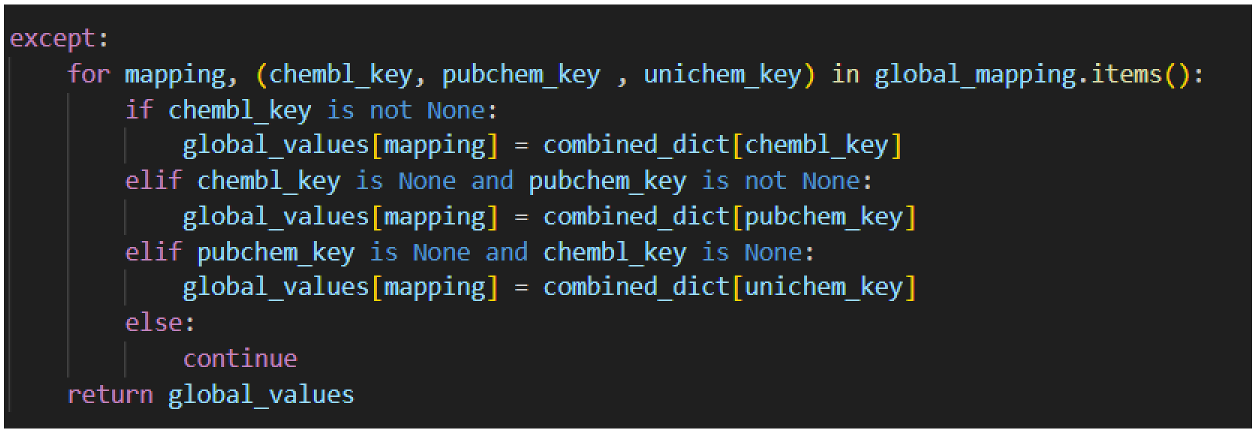
REMOVING UNICHEM DATA SOURCE

We will check whether the data source is present or not. If yes, then we will remove the source.



```python
elif source_add == 'NONE' and source_remove != "NONE":        ##we are removing a mew data source
    if source_remove in data_sources:
        data_sources.pop(source_remove)
        functions(data_sources)
        print(source_remove , 'is been removed')
    else:
        print("REQUIRED DATA SOURCE IS NOT PRESENT")
```

Again, the mapping will be the same as the first one, where the global schema will consist of only CHEMBL and PubChem.
Here to make a global schema, we will apply different Python subquery.



```
except:
    for mapping, (chembl_key, pubchem_key , unichem_key) in global_mapping.items():
        if chembl_key is not None:
            global_values[mapping] = combined_dict[chembl_key]
        elif chembl_key is None and pubchem_key is not None:
            global_values[mapping] = combined_dict[pubchem_key]
        elif pubchem_key is None and chembl_key is None:
            global_values[mapping] = combined_dict[unichem_key]
        else:
            continue
    return global_values
```

As this mapping of new columns, addition and removal is not suggested to be automated(discussed with the professor and a very tedious task). We preferred going by the manual approach of handling the mapping and matching of the attributes with our global definition performing the ETL.