# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

Copyright 2025 | Nghiên Cứu Thuốc | www.nghiencuuthuoc.com | Zalo: +84888999311 |

## 🧩 Topic: Phân tích pathway của target bằng UniProt/KEGG

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu Chembl 35 này. Với vai trò là chuyên gia nghiên cứu và phát triển dược học, tôi sẽ giúp bạn xây dựng quy trình phân tích hiệu quả, giải quyết các vấn đề kỹ thuật và đưa ra các ví dụ code minh họa.

**Tổng quan về dự án**

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu Chembl 35, kết hợp với thư viện Rdkit để hỗ trợ nghiên cứu và phát triển thuốc. Bạn đã thiết lập một quy trình làm việc theo chuẩn AIMLOps Template, sử dụng PostgreSQL để lưu trữ dữ liệu và Jupyter Notebook để phân tích.

**Phân tích và mô hình phân tích**

Mục tiêu chính của bạn là trích xuất thông tin hữu ích từ Chembl 35 để hỗ trợ quá trình phát triển thuốc. Dưới đây là một số hướng phân tích và mô hình tiềm năng bạn có thể áp dụng:

1. **Phân tích hoạt tính sinh học (Bioactivity Analysis):**
   * **Mô tả:** Phân tích này tập trung vào việc xác định các hợp chất có hoạt tính sinh học mong muốn đối với một mục tiêu cụ thể.
   * **Ứng dụng:** Xác định các ứng viên tiềm năng cho quá trình phát triển thuốc.
   * **Phương pháp:**
     + Sử dụng các truy vấn SQL để lọc dữ liệu hoạt tính (ví dụ: IC50, Ki) cho một mục tiêu cụ thể.
     + Sử dụng Rdkit để tính toán các thuộc tính hóa lý (ví dụ: MW, LogP) và fingerprint của các hợp chất.
     + Xây dựng mô hình QSAR (Quantitative Structure-Activity Relationship) để dự đoán hoạt tính dựa trên cấu trúc hóa học.
2. **Phân tích tương quan cấu trúc-hoạt tính (SAR Analysis):**
   * **Mô tả:** Phân tích này nhằm mục đích tìm ra mối liên hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng.
   * **Ứng dụng:** Tối ưu hóa cấu trúc của các hợp chất để cải thiện hoạt tính.
   * **Phương pháp:**
     + Sử dụng Rdkit để tạo ra các fingerprint (ví dụ: MACCS, ECFP) đại diện cho cấu trúc hóa học.
     + Sử dụng các phương pháp thống kê và học máy (ví dụ: Random Forest, Support Vector Machine) để tìm ra các fingerprint có liên quan đến hoạt tính.
3. **Phân tích tương đồng hóa học (Chemical Similarity Analysis):**
   * **Mô tả:** Phân tích này nhằm mục đích tìm kiếm các hợp chất có cấu trúc tương tự với một hợp chất đã biết và dự đoán hoạt tính của chúng.
   * **Ứng dụng:** Tìm kiếm các hợp chất mới có tiềm năng hoạt tính tương tự.
   * **Phương pháp:**
     + Sử dụng Rdkit để tính toán các fingerprint và so sánh sự tương đồng giữa các hợp chất.
     + Sử dụng các thuật toán tìm kiếm tương đồng (ví dụ: k-Nearest Neighbors) để tìm các hợp chất tương tự.
4. **Phân tích đa dạng hóa học (Chemical Diversity Analysis):**
   * **Mô tả:** Phân tích này nhằm mục đích đánh giá sự đa dạng của một tập hợp các hợp chất.
   * **Ứng dụng:** Lựa chọn các hợp chất đa dạng để sàng lọc nhằm tăng khả năng tìm ra các chất dẫn đầu mới.
   * **Phương pháp:**
     + Sử dụng Rdkit để tính toán các fingerprint và đo lường sự khác biệt giữa các hợp chất.
     + Sử dụng các thuật toán phân cụm (ví dụ: k-means, hierarchical clustering) để nhóm các hợp chất tương tự lại với nhau.
5. **Dự đoán tính chất ADMET (ADMET Prediction):**
   * **Mô tả:** Dự đoán các tính chất hấp thụ, phân phối, chuyển hóa, bài tiết và độc tính của các hợp chất.
   * **Ứng dụng:** Loại bỏ các hợp chất có tính chất ADMET không mong muốn ở giai đoạn sớm của quá trình phát triển thuốc.
   * **Phương pháp:**
     + Sử dụng Rdkit để tính toán các thuộc tính hóa lý và cấu trúc.
     + Sử dụng các mô hình học máy (ví dụ: Random Forest, Support Vector Machine) để dự đoán các tính chất ADMET.

**Hướng dẫn song ngữ và code mẫu**

Dưới đây là một số ví dụ về code SQL và Python, kèm theo giải thích song ngữ:

**1. Trích xuất dữ liệu hoạt tính (Extracting Activity Data):**

* **SQL (Tiếng Anh):**
* SELECT  
   cmp.chembl\_id,  
   act.standard\_type,  
   act.standard\_value,  
   act.standard\_units  
  FROM  
   activities act  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno  
  WHERE  
   act.standard\_type = 'IC50'  
   AND act.standard\_units = 'nM'  
  LIMIT 100;
  + **SQL (Tiếng Việt):**
* SELECT  
   cmp.chembl\_id, -- Mã Chembl của hợp chất  
   act.standard\_type, -- Loại hoạt tính (ví dụ: IC50)  
   act.standard\_value, -- Giá trị hoạt tính  
   act.standard\_units -- Đơn vị hoạt tính  
  FROM  
   activities act -- Bảng chứa thông tin hoạt tính  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno -- Bảng chứa thông tin hợp chất  
  WHERE  
   act.standard\_type = 'IC50' -- Lọc các hoạt tính IC50  
   AND act.standard\_units = 'nM' -- Lọc các hoạt tính có đơn vị nM  
  LIMIT 100; -- Giới hạn kết quả trả về 100 dòng
* **Python (Tiếng Anh):**
* import psycopg2  
  import pandas as pd  
  import os  
    
  # Database credentials  
  db\_params = {  
   'host': '192.168.206.136',  
   'user': 'rd',  
   'password': 'rd',  
   'database': 'chembl\_35'  
  }  
    
  # SQL query  
  sql\_query = """  
  SELECT  
   cmp.chembl\_id,  
   act.standard\_type,  
   act.standard\_value,  
   act.standard\_units  
  FROM  
   activities act  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno  
  WHERE  
   act.standard\_type = 'IC50'  
   AND act.standard\_units = 'nM'  
  LIMIT 100;  
  """  
    
  # Connect to the database and execute the query  
  try:  
   conn = psycopg2.connect(\*\*db\_params)  
   df = pd.read\_sql\_query(sql\_query, conn)  
   conn.close()  
   print("Data extracted successfully!")  
  except Exception as e:  
   print(f"Error: {e}")  
    
  # Print the first 5 rows of the DataFrame  
  print(df.head())  
    
  # Save the DataFrame to a CSV file  
  base\_path = "." # current directory  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df.to\_csv(csv\_file\_path, index=False)  
  print(f"Data saved to {csv\_file\_path}")
  + **Python (Tiếng Việt):**
* import psycopg2 # Thư viện kết nối PostgreSQL  
  import pandas as pd # Thư viện xử lý dữ liệu  
  import os # Thư viện tương tác với hệ điều hành  
    
  # Thông tin kết nối database  
  db\_params = {  
   'host': '192.168.206.136',  
   'user': 'rd',  
   'password': 'rd',  
   'database': 'chembl\_35'  
  }  
    
  # Câu truy vấn SQL  
  sql\_query = """  
  SELECT  
   cmp.chembl\_id, -- Mã Chembl của hợp chất  
   act.standard\_type, -- Loại hoạt tính (ví dụ: IC50)  
   act.standard\_value, -- Giá trị hoạt tính  
   act.standard\_units -- Đơn vị hoạt tính  
  FROM  
   activities act -- Bảng chứa thông tin hoạt tính  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno -- Bảng chứa thông tin hợp chất  
  WHERE  
   act.standard\_type = 'IC50' -- Lọc các hoạt tính IC50  
   AND act.standard\_units = 'nM' -- Lọc các hoạt tính có đơn vị nM  
  LIMIT 100; -- Giới hạn kết quả trả về 100 dòng  
  """  
    
  # Kết nối database và thực hiện truy vấn  
  try:  
   conn = psycopg2.connect(\*\*db\_params)  
   df = pd.read\_sql\_query(sql\_query, conn) # Đọc dữ liệu vào DataFrame  
   conn.close() # Đóng kết nối  
   print("Dữ liệu đã được trích xuất thành công!")  
  except Exception as e:  
   print(f"Lỗi: {e}")  
    
  # In 5 dòng đầu tiên của DataFrame  
  print(df.head())  
    
  # Lưu DataFrame vào file CSV  
  base\_path = "." # thư mục hiện tại  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df.to\_csv(csv\_file\_path, index=False)  
  print(f"Dữ liệu đã được lưu vào {csv\_file\_path}")

**2. Tính toán fingerprint với Rdkit (Calculating Fingerprints with Rdkit):**

* **Python (Tiếng Anh):**
* from rdkit import Chem  
  from rdkit.Chem import AllChem  
  import pandas as pd  
  import os  
    
  # Load data from CSV file  
  base\_path = "." # current directory  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
  # Function to calculate Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles)  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
   return fp.ToBitString()  
   else:  
   return None  
   except:  
   return None  
    
  # Assuming you have a 'smiles' column in your DataFrame  
  # Example SMILES (replace with your actual SMILES column)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
    
  # Create a DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list})  
    
  # Apply the function to create a new column 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # Print the DataFrame with Morgan fingerprints  
  print(example\_df)
  + **Python (Tiếng Việt):**
* from rdkit import Chem # Thư viện Rdkit  
  from rdkit.Chem import AllChem  
  import pandas as pd # Thư viện xử lý dữ liệu  
  import os  
    
  # Load dữ liệu từ file CSV  
  base\_path = "." # thư mục hiện tại  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
  # Hàm tính toán Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles) # Chuyển SMILES thành đối tượng molecule  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048) # Tính toán Morgan fingerprint  
   return fp.ToBitString() # Chuyển fingerprint thành chuỗi bit  
   else:  
   return None  
   except:  
   return None  
    
  # Giả sử bạn có cột 'smiles' trong DataFrame  
  # Ví dụ danh sách SMILES (thay thế bằng cột SMILES thực tế của bạn)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
    
  # Tạo DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list})  
    
  # Áp dụng hàm để tạo cột mới 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # In DataFrame với Morgan fingerprints  
  print(example\_df)

**3. Phân tích tương đồng hóa học (Chemical Similarity Analysis):**

* **Python (Tiếng Anh):**
* from rdkit import Chem  
  from rdkit.Chem import AllChem  
  from rdkit.DataStructs import DiceSimilarity  
  import pandas as pd  
  import os  
    
  # Load data from CSV file  
  base\_path = "." # current directory  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
    
  # Function to calculate Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles)  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
   return fp  
   else:  
   return None  
   except:  
   return None  
    
  # Assuming you have a 'smiles' column in your DataFrame  
  # Example SMILES (replace with your actual SMILES column)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
    
  # Create a DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list})  
    
  # Apply the function to create a new column 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # Function to calculate Dice similarity  
  def calculate\_dice\_similarity(fp1, fp2):  
   if fp1 is None or fp2 is None:  
   return None  
   return DiceSimilarity(fp1, fp2)  
    
  # Calculate similarity between the first compound and the rest  
  first\_fingerprint = example\_df['morgan\_fingerprint'][0]  
  example\_df['similarity'] = example\_df['morgan\_fingerprint'].apply(lambda fp: calculate\_dice\_similarity(first\_fingerprint, fp))  
    
  # Print the DataFrame with similarity scores  
  print(example\_df)
  + **Python (Tiếng Việt):**
* from rdkit import Chem # Thư viện Rdkit  
  from rdkit.Chem import AllChem  
  from rdkit.DataStructs import DiceSimilarity  
  import pandas as pd # Thư viện xử lý dữ liệu  
  import os  
    
  # Load dữ liệu từ file CSV  
  base\_path = "." # thư mục hiện tại  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
  # Hàm tính toán Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles) # Chuyển SMILES thành đối tượng molecule  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048) # Tính toán Morgan fingerprint  
   return fp  
   else:  
   return None  
   except:  
   return None  
    
  # Giả sử bạn có cột 'smiles' trong DataFrame  
  # Ví dụ danh sách SMILES (thay thế bằng cột SMILES thực tế của bạn)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
    
  # Tạo DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list})  
    
  # Áp dụng hàm để tạo cột mới 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # Hàm tính toán độ tương đồng Dice  
  def calculate\_dice\_similarity(fp1, fp2):  
   if fp1 is None or fp2 is None:  
   return None  
   return DiceSimilarity(fp1, fp2)  
    
  # Tính toán độ tương đồng giữa hợp chất đầu tiên và các hợp chất còn lại  
  first\_fingerprint = example\_df['morgan\_fingerprint'][0]  
  example\_df['similarity'] = example\_df['morgan\_fingerprint'].apply(lambda fp: calculate\_dice\_similarity(first\_fingerprint, fp))  
    
  # In DataFrame với độ tương đồng  
  print(example\_df)

**4. Mô hình QSAR đơn giản (Simple QSAR Modeling):**

* **Python (Tiếng Anh):**
* from rdkit import Chem  
  from rdkit.Chem import AllChem  
  import pandas as pd  
  from sklearn.model\_selection import train\_test\_split  
  from sklearn.linear\_model import LinearRegression  
  from sklearn.metrics import mean\_squared\_error  
  import os  
    
  # Load data from CSV file  
  base\_path = "." # current directory  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
  # Function to calculate Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles)  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
   return fp  
   else:  
   return None  
   except:  
   return None  
    
  # Assuming you have a 'smiles' column and a 'standard\_value' column in your DataFrame  
  # Example SMILES (replace with your actual SMILES column)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
  activity\_list = [5, 2, 1]  
    
  # Create a DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list, 'standard\_value': activity\_list})  
    
  # Apply the function to create a new column 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # Convert fingerprints to lists of integers  
  example\_df['fingerprint\_list'] = example\_df['morgan\_fingerprint'].apply(lambda fp: list(map(int, fp.ToBitString())))  
    
  # Drop rows with None in 'morgan\_fingerprint'  
  example\_df = example\_df.dropna(subset=['morgan\_fingerprint'])  
    
  # Prepare data for the model  
  X = list(example\_df['fingerprint\_list'])  
  y = example\_df['standard\_value']  
    
  # Split data into training and testing sets  
  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
    
  # Train a linear regression model  
  model = LinearRegression()  
  model.fit(X\_train, y\_train)  
    
  # Make predictions on the test set  
  y\_pred = model.predict(X\_test)  
    
  # Evaluate the model  
  mse = mean\_squared\_error(y\_test, y\_pred) # squared=False is not needed for the newest version  
  print(f"Mean Squared Error: {mse}")
  + **Python (Tiếng Việt):**
* from rdkit import Chem # Thư viện Rdkit  
  from rdkit.Chem import AllChem  
  import pandas as pd # Thư viện xử lý dữ liệu  
  from sklearn.model\_selection import train\_test\_split  
  from sklearn.linear\_model import LinearRegression  
  from sklearn.metrics import mean\_squared\_error  
  import os  
    
  # Load dữ liệu từ file CSV  
  base\_path = "." # thư mục hiện tại  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_activity\_data.csv")  
  df = pd.read\_csv(csv\_file\_path)  
    
  # Hàm tính toán Morgan fingerprint  
  def calculate\_morgan\_fingerprint(smiles):  
   try:  
   mol = Chem.MolFromSmiles(smiles) # Chuyển SMILES thành đối tượng molecule  
   if mol:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048) # Tính toán Morgan fingerprint  
   return fp  
   else:  
   return None  
   except:  
   return None  
    
  # Giả sử bạn có cột 'smiles' và cột 'standard\_value' trong DataFrame  
  # Ví dụ danh sách SMILES (thay thế bằng cột SMILES thực tế của bạn)  
  smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
  activity\_list = [5, 2, 1]  
    
  # Tạo DataFrame  
  example\_df = pd.DataFrame({'smiles': smiles\_list, 'standard\_value': activity\_list})  
    
  # Áp dụng hàm để tạo cột mới 'morgan\_fingerprint'  
  example\_df['morgan\_fingerprint'] = example\_df['smiles'].apply(calculate\_morgan\_fingerprint)  
    
  # Chuyển fingerprints thành danh sách các số nguyên  
  example\_df['fingerprint\_list'] = example\_df['morgan\_fingerprint'].apply(lambda fp: list(map(int, fp.ToBitString())))  
    
  # Loại bỏ các hàng có giá trị None trong cột 'morgan\_fingerprint'  
  example\_df = example\_df.dropna(subset=['morgan\_fingerprint'])  
    
  # Chuẩn bị dữ liệu cho mô hình  
  X = list(example\_df['fingerprint\_list'])  
  y = example\_df['standard\_value']  
    
  # Chia dữ liệu thành tập huấn luyện và tập kiểm tra  
  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
    
  # Huấn luyện mô hình hồi quy tuyến tính  
  model = LinearRegression()  
  model.fit(X\_train, y\_train)  
    
  # Dự đoán trên tập kiểm tra  
  y\_pred = model.predict(X\_test)  
    
  # Đánh giá mô hình  
  mse = mean\_squared\_error(y\_test, y\_pred) # squared=False is not needed for the newest version  
  print(f"Mean Squared Error: {mse}")

**5. Lọc dữ liệu theo khoảng giá trị (Filtering data by range):**

* **SQL (Tiếng Anh):**
* SELECT  
   cmp.chembl\_id,  
   act.standard\_value  
  FROM  
   activities act  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno  
  WHERE  
   act.standard\_type = 'IC50'  
   AND act.standard\_units = 'nM'  
   AND act.standard\_value BETWEEN 100 AND 1000  
  LIMIT 100;
  + **SQL (Tiếng Việt):**
* SELECT  
   cmp.chembl\_id, -- Mã Chembl của hợp chất  
   act.standard\_value -- Giá trị hoạt tính  
  FROM  
   activities act -- Bảng chứa thông tin hoạt tính  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno -- Bảng chứa thông tin hợp chất  
  WHERE  
   act.standard\_type = 'IC50' -- Lọc các hoạt tính IC50  
   AND act.standard\_units = 'nM' -- Lọc các hoạt tính có đơn vị nM  
   AND act.standard\_value BETWEEN 100 AND 1000 -- Lọc các giá trị trong khoảng 100-1000  
  LIMIT 100; -- Giới hạn kết quả trả về 100 dòng
* **Python (Tiếng Anh):**
* import psycopg2  
  import pandas as pd  
  import os  
    
  # Database credentials  
  db\_params = {  
   'host': '192.168.206.136',  
   'user': 'rd',  
   'password': 'rd',  
   'database': 'chembl\_35'  
  }  
    
  # SQL query  
  sql\_query = """  
  SELECT  
   cmp.chembl\_id,  
   act.standard\_value  
  FROM  
   activities act  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno  
  WHERE  
   act.standard\_type = 'IC50'  
   AND act.standard\_units = 'nM'  
   AND act.standard\_value BETWEEN 100 AND 1000  
  LIMIT 100;  
  """  
    
  # Connect to the database and execute the query  
  try:  
   conn = psycopg2.connect(\*\*db\_params)  
   df = pd.read\_sql\_query(sql\_query, conn)  
   conn.close()  
   print("Data extracted successfully!")  
  except Exception as e:  
   print(f"Error: {e}")  
    
  # Print the first 5 rows of the DataFrame  
  print(df.head())  
    
  # Save the DataFrame to a CSV file  
  base\_path = "." # current directory  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_filtered\_data.csv")  
  df.to\_csv(csv\_file\_path, index=False)  
  print(f"Data saved to {csv\_file\_path}")
  + **Python (Tiếng Việt):**
* import psycopg2 # Thư viện kết nối PostgreSQL  
  import pandas as pd # Thư viện xử lý dữ liệu  
  import os # Thư viện tương tác với hệ điều hành  
    
  # Thông tin kết nối database  
  db\_params = {  
   'host': '192.168.206.136',  
   'user': 'rd',  
   'password': 'rd',  
   'database': 'chembl\_35'  
  }  
    
  # Câu truy vấn SQL  
  sql\_query = """  
  SELECT  
   cmp.chembl\_id, -- Mã Chembl của hợp chất  
   act.standard\_value -- Giá trị hoạt tính  
  FROM  
   activities act -- Bảng chứa thông tin hoạt tính  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno -- Bảng chứa thông tin hợp chất  
  WHERE  
   act.standard\_type = 'IC50' -- Lọc các hoạt tính IC50  
   AND act.standard\_units = 'nM' -- Lọc các hoạt tính có đơn vị nM  
   AND act.standard\_value BETWEEN 100 AND 1000 -- Lọc các giá trị trong khoảng 100-1000  
  LIMIT 100; -- Giới hạn kết quả trả về 100 dòng  
  """  
    
  # Kết nối database và thực hiện truy vấn  
  try:  
   conn = psycopg2.connect(\*\*db\_params)  
   df = pd.read\_sql\_query(sql\_query, conn) # Đọc dữ liệu vào DataFrame  
   conn.close() # Đóng kết nối  
   print("Dữ liệu đã được trích xuất thành công!")  
  except Exception as e:  
   print(f"Lỗi: {e}")  
    
  # In 5 dòng đầu tiên của DataFrame  
  print(df.head())  
    
  # Lưu DataFrame vào file CSV  
  base\_path = "." # thư mục hiện tại  
  csv\_file\_path = os.path.join(base\_path, "Topic\_CheMBL\_35\_29\_1\_filtered\_data.csv")  
  df.to\_csv(csv\_file\_path, index=False)  
  print(f"Dữ liệu đã được lưu vào {csv\_file\_path}")

**Giải quyết lỗi**

* **Lỗi a: ERROR: operator does not exist: numeric ~ unknown**
* Lỗi này xảy ra do bạn đang cố gắng so sánh một cột kiểu số (numeric) với một chuỗi (unknown). Để khắc phục, bạn cần đảm bảo rằng cột act.standard\_value chỉ chứa các giá trị số hợp lệ. Bạn có thể sử dụng hàm REGEXP\_MATCHES để kiểm tra xem giá trị có phải là số hay không trước khi so sánh:
* SELECT  
   cmp.chembl\_id,  
   act.standard\_type,  
   act.standard\_value,  
   act.standard\_units  
  FROM  
   activities act  
  JOIN  
   molecule\_dictionary cmp ON act.molregno = cmp.molregno  
  WHERE  
   act.standard\_type = 'IC50'  
   AND act.standard\_units = 'nM'  
   AND act.standard\_value::text SIMILAR TO '[0-9\.]+'  
  LIMIT 100;
* **Explanation (Tiếng Anh):**
* The error occurs because you are trying to compare a numeric column with a string. To fix this, ensure that the act.standard\_value column contains only valid numeric values. Use SIMILAR TO to check if the value is a number before comparison.
* **Explanation (Tiếng Việt):**
* Lỗi xảy ra do bạn đang cố gắng so sánh một cột kiểu số với một chuỗi. Để khắc phục, hãy đảm bảo rằng cột act.standard\_value chỉ chứa các giá trị số hợp lệ. Sử dụng SIMILAR TO để kiểm tra xem giá trị có phải là số hay không trước khi so sánh.
* **Lỗi b: phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean\_squared\_error**
* Nếu bạn đang sử dụng phiên bản scikit-learn cũ, hãy loại bỏ tham số squared=False khỏi hàm mean\_squared\_error. Nếu có thể, hãy nâng cấp lên phiên bản scikit-learn mới nhất để sử dụng tham số này.
* **Explanation (Tiếng Anh):**
* If you are using an older version of scikit-learn, remove the squared=False parameter from the mean\_squared\_error function. If possible, upgrade to the latest version of scikit-learn to use this parameter.
* **Explanation (Tiếng Việt):**
* Nếu bạn đang sử dụng phiên bản scikit-learn cũ, hãy loại bỏ tham số squared=False khỏi hàm mean\_squared\_error. Nếu có thể, hãy nâng cấp lên phiên bản scikit-learn mới nhất để sử dụng tham số này.

**Lưu ý quan trọng:**

* **Đường dẫn file:** Luôn sử dụng os.path.join(base\_path, "your\_file.csv") để đảm bảo tính tương thích giữa các hệ điều hành.
* **Xử lý lỗi:** Luôn sử dụng try...except để xử lý các lỗi có thể xảy ra trong quá trình thực thi code.
* **Kiểm tra dữ liệu:** Luôn kiểm tra dữ liệu sau khi trích xuất và tiền xử lý để đảm bảo tính chính xác.

Chúc bạn thành công với dự án của mình! Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi.