

Topic: Comparing ChEMBL drugs with WHO Essential Medicines

Okay, I understand. You're working on a ChEMBL 35 data analysis project using RDKit, following an AIMLops template, and need help structuring your SQL queries and Python code. Let's address your error messages and structure your workflow with examples.

1. Analysis Model:

The project's core involves extracting and analyzing data from the ChEMBL 35 database. Here's a breakdown of the analysis model:

- **Data Extraction (SQL):** The initial step involves querying the ChEMBL 35 database using SQL to extract relevant information about bioactivities, compounds, and targets. This will involve joining tables like `activities`, `molecule_dictionary`, `compound_structures`, etc., based on foreign keys. We'll aim to retrieve a limited number of rows (100) to minimize processing time.
- **Data Transformation (Python/RDKit):** After extracting the data, we'll use Python and RDKit to transform it into a suitable format for analysis. This might include:
 - Converting SMILES strings to RDKit Mol objects.
 - Calculating molecular descriptors (e.g., molecular weight, LogP, number of hydrogen bond donors/acceptors).
 - Handling missing data or outliers.
- **Data Analysis (Python/SciKit-Learn/Other Libraries):** This is where the core analysis happens. Based on the specific "topic" you are investigating, this could involve:
 - Building predictive models for activity (e.g., classification or regression).
 - Analyzing structure-activity relationships (SAR).
 - Clustering compounds based on their properties.
 - Performing statistical analysis to identify significant features.
- **Error Handling and Version Compatibility:** Your error messages indicate issues with:
 - **SQL String Matching:** You're trying to use the `~` operator (regular expression matching) on a numeric column. This will need correction.
 - **Scikit-Learn Version:** You'll need to either upgrade your Scikit-Learn version or adjust the `mean_squared_error` call to be compatible with your installed version.

2. SQL Code (English & Vietnamese):

Here's an example SQL query tailored to extract relevant data, addressing the reported error, and limiting the output to 100 rows. This example focuses on retrieving data related to a specific target and filtering activities based on the standard type.

-- English

SELECT

```
act.activity_id,  
md.chembl_id,  
cs.canonical_smiles,  
act.standard_type,  
act.standard_value,  
act.standard_units,  
t.target_chembl_id
```

```

FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    target_dictionary t ON act.tid = t.tid
WHERE
    t.target_chembl_id = 'CHEMBL205' -- Example target
    AND act.standard_type = 'IC50' -- Example activity type
    AND act.standard_value IS NOT NULL -- Ensure standard_value is not null
    AND act.standard_value::TEXT SIMILAR TO '[0-9.]+' -- Cast to text and use SIMILAR
TO
LIMIT 100;

-- Vietnamese

-- Lựa chọn các cột cần thiết từ các bảng khác nhau
SELECT
    act.activity_id, -- ID của hoạt động sinh học
    md.chembl_id, -- ID ChEMBL của phân tử
    cs.canonical_smiles, -- Cấu trúc SMILES của phân tử
    act.standard_type, -- Loại giá trị chuẩn (ví dụ: IC50)
    act.standard_value, -- Giá trị chuẩn
    act.standard_units, -- Đơn vị của giá trị chuẩn
    t.target_chembl_id -- ID ChEMBL của mục tiêu
FROM
    activities act -- Bảng hoạt động sinh học
JOIN
    molecule_dictionary md ON act.molregno = md.molregno -- Kết nối với bảng thông
tin phân tử
JOIN
    compound_structures cs ON md.molregno = cs.molregno -- Kết nối với bảng cấu trúc
phân tử
JOIN
    target_dictionary t ON act.tid = t.tid -- Kết nối với bảng thông tin mục tiêu
WHERE
    t.target_chembl_id = 'CHEMBL205' -- Lọc theo ID mục tiêu (ví dụ)
    AND act.standard_type = 'IC50' -- Lọc theo loại hoạt động (ví dụ)
    AND act.standard_value IS NOT NULL -- Đảm bảo giá trị chuẩn không rỗng
    AND act.standard_value::TEXT SIMILAR TO '[0-9.]+' -- Chuyển đổi sang text và sử
dụng SIMILAR TO để so khớp
LIMIT 100; -- Giới hạn số lượng kết quả trả về là 100

```

Explanation of SQL Correction:

- **AND act.standard_value::TEXT SIMILAR TO '[0-9.]+'**: The key change is casting `act.standard_value` to `TEXT` using `::TEXT` before using `SIMILAR TO`. This ensures that the regular expression matching works correctly on a string representation of the numeric value. `SIMILAR TO` is a SQL standard compliant version of Regular Expression matching.

How to Run the SQL:

1. Open pgAdmin and connect to your chembl_35 database (ip: 192.168.206.136, user: rd, pass: rd).
2. Create a new query window.
3. Paste the SQL code into the query window.
4. Execute the query.

5. Save the results as a CSV file (e.g., ../data/chembl_data.csv). You can usually do this directly from pgAdmin's query result window by right-clicking and selecting "Copy with Headers" or "Save as CSV."

3. Python Code (English & Vietnamese):

English

```
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

import os

# Define base path for AIMLops standard
base_path = "../" # Adjust if your base path is different
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Load the CSV file
data_file = os.path.join(data_path, "chembl_data.csv") # Construct complete path
df = pd.read_csv(data_file)

# Handle missing values by filling with the mean.
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df['standard_value'] = df['standard_value'].fillna(df['standard_value'].mean())

# Convert SMILES to RDKit Mol objects
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Calculate molecular weight
df['mol_weight'] = df['mol'].apply(lambda x: Descriptors.MolWt(x) if x else None)

# Drop rows with missing Mol objects (invalid SMILES)
df = df.dropna(subset=['mol'])

# Prepare data for modeling
X = df[['mol_weight']] # Features
y = df['standard_value'] # Target variable

# 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)

# Impute missing values with the mean (after splitting the data!)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())

# Train a Linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
```

```

try:
    mse = mean_squared_error(y_test, y_pred, squared=False) # or squared=True,
    depending on version
except TypeError as e:
    print(f"TypeError: {e}")
    print("It seems that your scikit-learn version does not support squared=False, try
    removing the squared parameter or updating the sklearn package")
    mse = mean_squared_error(y_test, y_pred) # Older version compatibility

print(f"Mean Squared Error: {mse}")

# Vietnamese
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

import os

# Định nghĩa đường dẫn gốc theo chuẩn AIMLops
base_path = "../" # Điều chỉnh nếu đường dẫn gốc của bạn khác
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Tải tệp CSV
data_file = os.path.join(data_path, "chembl_data.csv") # Xây dựng đường dẫn đầy đủ
df = pd.read_csv(data_file)

# Xử lý giá trị thiếu bằng cách điền vào bằng giá trị trung bình.
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df['standard_value'] = df['standard_value'].fillna(df['standard_value'].mean())

# Chuyển đổi chuỗi SMILES thành đối tượng Mol của RDKit
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Tính toán khối lượng phân tử
df['mol_weight'] = df['mol'].apply(lambda x: Descriptors.MolWt(x) if x else None)

# Loại bỏ các hàng có đối tượng Mol bị thiếu (SMILES không hợp lệ)
df = df.dropna(subset=['mol'])

# Chuẩn bị dữ liệu cho mô hình hóa
X = df[['mol_weight']] # Các đặc trưng
y = df['standard_value'] # Biến mục tiêu

# 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)

# Điền giá trị thiếu bằng giá trị trung bình (sau khi chia dữ liệu!)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())

# Huấn luyện mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

```

```
# Dự đoán
y_pred = model.predict(X_test)

# Đánh giá mô hình
try:
    mse = mean_squared_error(y_test, y_pred, squared=False) # hoặc squared=True, tùy
    thuộc vào phiên bản
except TypeError as e:
    print(f"TypeError: {e}")
    print("Có vẻ như phiên bản scikit-learn của bạn không hỗ trợ squared=False, hãy
    thử xóa tham số squared hoặc cập nhật gói sklearn")
    mse = mean_squared_error(y_test, y_pred) # Tương thích với phiên bản cũ hơn

print(f"Sai số bình phương trung bình: {mse}")
```

Explanation of Python Code:

1. **Import Libraries:** Imports necessary libraries like pandas, RDKit, and scikit-learn.
2. **AIMLops Path Handling:** Uses `os.path.join` to construct file paths according to your AIMLops template. This is crucial for reproducibility and maintainability.
3. **Data Loading:** Loads the CSV file you created from the SQL query.
4. **SMILES Conversion:** Converts SMILES strings to RDKit Mol objects, which are necessary for calculating molecular descriptors. Handles potential errors by skipping invalid SMILES.
5. **Descriptor Calculation:** Calculates a simple molecular descriptor (molecular weight). You can add more descriptors as needed.
6. **Data Preparation:** Selects the feature(s) (X) and the target variable (y).
7. **Train/Test Split:** Splits the data into training and testing sets.
8. **Model Training:** Trains a linear regression model.
9. **Prediction and Evaluation:** Makes predictions on the test set and evaluates the model using Mean Squared Error (MSE).
10. **Error Handling:** Includes a `try...except` block to handle the `TypeError` related to the `squared` parameter in `mean_squared_error`.

How to Run the Python Code:

1. Create a new Jupyter Notebook file named `Topic_CheMBL_35_56_1_analysis.ipynb` in your `../notebooks` directory.
2. Paste the Python code into a cell in the notebook.
3. Run the cell. Make sure the CSV file (`chembl_data.csv`) is in the correct location (`../data`).

4. Examples (Based on Topic_CheMBL_35_56):

Since you haven't provided the specific details of "Topic_CheMBL_35_56", I'll create 5 general examples of analyses that could be performed with ChEMBL 35 data, along with modifications to the Python code. **You'll need to adapt these to your specific research question.**

Example 1: Predicting IC50 Values from Molecular Weight

- **Description:** This is the example already implemented in the code above. It's a basic regression model using molecular weight to predict IC50 values.
- **Code Modifications:** No changes needed to the Python code provided.

Example 2: Predicting Activity (Active/Inactive) based on LogP

- **Description:** Change the analysis to predict a binary activity label (active/inactive) based on the calculated LogP value.
- **SQL Modifications:** (Add `standard_value` to SQL Query)

```

SELECT
    act.activity_id,
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    t.target_chembl_id
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    target_dictionary t ON act.tid = t.tid
WHERE
    t.target_chembl_id = 'CHEMBL205' -- Example target
    AND act.standard_type = 'IC50' -- Example activity type
    AND act.standard_value IS NOT NULL -- Ensure standard_value is not null
    AND act.standard_value::TEXT SIMILAR TO '[0-9.]+' -- Cast to text and use SIMILAR
TO
LIMIT 100;

```

- **Python Code Modifications:**

```

# English
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression # Changed model
from sklearn.metrics import accuracy_score, confusion_matrix # Changed metrics

import os

# Define base path for AIMLops standard
base_path = "../" # Adjust if your base path is different
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Load the CSV file
data_file = os.path.join(data_path, "chembl_data.csv") # Construct complete path
df = pd.read_csv(data_file)

# Define activity threshold
activity_threshold = 10000 # Example threshold for IC50 (nM)

# Create binary activity label
df['active'] = (df['standard_value'] <= activity_threshold).astype(int)

# Convert SMILES to RDKit Mol objects
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Calculate LogP
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)

# Drop rows with missing Mol objects or LogP values
df = df.dropna(subset=['mol', 'logp'])

```

```

# Prepare data for modeling
X = df[['logp']] # Feature is now LogP
y = df['active'] # Target is now binary activity

# 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)

# Impute missing values with the mean (after splitting the data!)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())

# Train a Logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")

# Vietnamese
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression # Đã thay đổi mô hình
from sklearn.metrics import accuracy_score, confusion_matrix # Đã thay đổi độ đo

import os

# Định nghĩa đường dẫn gốc theo chuẩn AIMLops
base_path = "../" # Điều chỉnh nếu đường dẫn gốc của bạn khác
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Tải tệp CSV
data_file = os.path.join(data_path, "chembl_data.csv") # Xây dựng đường dẫn đầy đủ
df = pd.read_csv(data_file)

# Xác định ngưỡng hoạt động
activity_threshold = 10000 # Ví dụ: ngưỡng cho IC50 (nM)

# Tạo nhãn hoạt động nhị phân
df['active'] = (df['standard_value'] <= activity_threshold).astype(int)

# Chuyển đổi chuỗi SMILES thành đối tượng Mol của RDKit
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Tính toán LogP
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)

```



```

# Loại bỏ các hàng có đối tượng Mol hoặc giá trị LogP bị thiếu
df = df.dropna(subset=['mol', 'logp'])

# Chuẩn bị dữ liệu cho mô hình hóa
X = df[['logp']] # Đặc trưng bây giờ là LogP
y = df['active'] # Biến mục tiêu bây giờ là hoạt động nhị phân

# 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)

# Điền giá trị thiếu bằng giá trị trung bình (sau khi chia dữ liệu!)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())

# Huấn luyện mô hình hồi quy Logistic
model = LogisticRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)

# Đánh giá mô hình
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print(f"Độ chính xác: {accuracy}")
print(f"Ma trận nhầm lẫn:\n{confusion}")

```

Example 3: Clustering Compounds based on Molecular Descriptors

- **Description:** Cluster compounds based on multiple molecular descriptors (e.g., molecular weight, LogP, number of hydrogen bond donors/acceptors).
- **Python Code Modifications:**

```

# English
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.cluster import KMeans # Changed model
from sklearn.preprocessing import StandardScaler # For scaling features
import os

# Define base path for AIMLops standard
base_path = "../" # Adjust if your base path is different
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Load the CSV file
data_file = os.path.join(data_path, "chembl_data.csv") # Construct complete path
df = pd.read_csv(data_file)

# Convert SMILES to RDKit Mol objects
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Calculate molecular descriptors
df['mol_weight'] = df['mol'].apply(lambda x: Descriptors.MolWt(x) if x else None)
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)

```



```

df['hbd'] = df['mol'].apply(lambda x: Descriptors.NumHDonors(x) if x else None) #
Hydrogen bond donors
df['hba'] = df['mol'].apply(lambda x: Descriptors.NumHAcceptors(x) if x else None) #
Hydrogen bond acceptors

# Drop rows with missing Mol objects or descriptor values
df = df.dropna(subset=['mol', 'mol_weight', 'logp', 'hbd', 'hba'])

# Prepare data for clustering
X = df[['mol_weight', 'logp', 'hbd', 'hba']] # Multiple features

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Determine the optimal number of clusters (Elbow method - not shown in code for
brevity)
# You'd typically plot the within-cluster sum of squares for different numbers of
clusters
# and choose the "elbow" point. Let's assume k=3.
n_clusters = 3

# Perform K-means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init = 'auto') #Added n_init
df['cluster'] = kmeans.fit_predict(X_scaled)

# Analyze the clusters (e.g., calculate mean descriptor values for each cluster)
cluster_means = df.groupby('cluster')[['mol_weight', 'logp', 'hbd', 'hba']].mean()
print(cluster_means)

# Vietnamese
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.cluster import KMeans # Đã thay đổi mô hình
from sklearn.preprocessing import StandardScaler # Để chuẩn hóa các đặc trưng
import os

# Định nghĩa đường dẫn gốc theo chuẩn AIMLops
base_path = "../" # Điều chỉnh nếu đường dẫn gốc của bạn khác
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Tải tệp CSV
data_file = os.path.join(data_path, "chembl_data.csv") # Xây dựng đường dẫn đầy đủ
df = pd.read_csv(data_file)

# Chuyển đổi chuỗi SMILES thành đối tượng Mol của RDKit
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Tính toán các đặc trưng phân tử
df['mol_weight'] = df['mol'].apply(lambda x: Descriptors.MolWt(x) if x else None)
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)
df['hbd'] = df['mol'].apply(lambda x: Descriptors.NumHDonors(x) if x else None) # Số
Lượng liên kết hydro cho
df['hba'] = df['mol'].apply(lambda x: Descriptors.NumHAcceptors(x) if x else None) #
Số Lượng liên kết hydro nhận

```

```

# Loại bỏ các hàng có đối tượng Mol hoặc giá trị đặc trưng bị thiếu
df = df.dropna(subset=['mol', 'mol_weight', 'logp', 'hbd', 'hba'])

# Chuẩn bị dữ liệu cho phân cụm
X = df[['mol_weight', 'logp', 'hbd', 'hba']] # Nhiều đặc trưng

# Chuẩn hóa các đặc trưng
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Xác định số lượng cụm tối ưu (Phương pháp khuỷu tay - không hiển thị trong mã để ngắn gọn)
# Thông thường, bạn sẽ vẽ tổng bình phương trong cụm cho các số lượng cụm khác nhau
# và chọn điểm "khuỷu tay". Giả sử k=3.
n_clusters = 3

# Thực hiện phân cụm K-means
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init = 'auto') # Added n_init
df['cluster'] = kmeans.fit_predict(X_scaled)

# Phân tích các cụm (ví dụ: tính giá trị trung bình của các đặc trưng cho mỗi cụm)
cluster_means = df.groupby('cluster')[['mol_weight', 'logp', 'hbd', 'hba']].mean()
print(cluster_means)

```

Example 4: Analyzing Structure-Activity Relationships (SAR) with Matplotlib

- **Description:** Visualize the relationship between a molecular descriptor (e.g., LogP) and activity (e.g., IC50) using a scatter plot. This helps identify trends in SAR.
- **Python Code Modifications:**

```

# English
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
import matplotlib.pyplot as plt # Added for plotting
import os

# Define base path for AIMLops standard
base_path = "../" # Adjust if your base path is different
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Load the CSV file
data_file = os.path.join(data_path, "chembl_data.csv") # Construct complete path
df = pd.read_csv(data_file)

# Convert SMILES to RDKit Mol objects
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Calculate LogP
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)

# Drop rows with missing Mol objects or LogP values
df = df.dropna(subset=['mol', 'logp', 'standard_value']) # Also drop missing standard_value

```

```

# Create the scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df['logp'], df['standard_value'], alpha=0.5)
plt.xlabel("LogP")
plt.ylabel("IC50 (standard_value)")
plt.title("Structure-Activity Relationship")
plt.yscale('log') # Often useful for IC50 values
plt.grid(True)
plt.show()

# Vietnamese
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
import matplotlib.pyplot as plt # Đã thêm để vẽ đồ thị
import os

# Định nghĩa đường dẫn gốc theo chuẩn AIMLops
base_path = "../" # Điều chỉnh nếu đường dẫn gốc của bạn khác
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Tải tệp CSV
data_file = os.path.join(data_path, "chembl_data.csv") # Xây dựng đường dẫn đầy đủ
df = pd.read_csv(data_file)

# Chuyển đổi chuỗi SMILES thành đối tượng Mol của RDKit
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))

# Tính toán LogP
df['logp'] = df['mol'].apply(lambda x: Descriptors.MolLogP(x) if x else None)

# Loại bỏ các hàng có đối tượng Mol hoặc giá trị LogP bị thiếu
df = df.dropna(subset=['mol', 'logp', 'standard_value']) # Cũng loại bỏ giá trị
standard_value bị thiếu

# Tạo biểu đồ phân tán
plt.figure(figsize=(8, 6))
plt.scatter(df['logp'], df['standard_value'], alpha=0.5)
plt.xlabel("LogP")
plt.ylabel("IC50 (standard_value)")
plt.title("Mối quan hệ Cấu trúc-Hoạt tính")
plt.yscale('log') # Thường hữu ích cho các giá trị IC50
plt.grid(True)
plt.show()

```

Example 5: Target Specific Activity Analysis

- **Description:** Focus your analysis on activities against a specific target (e.g., ChEMBL205). You might investigate which compounds are most potent against that target or build a model to predict activity specifically for that target.
- **SQL Modifications:** (filter target)

-- English

```

SELECT
    act.activity_id,
    md.chembl_id,

```

```

    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    t.target_chembl_id
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    target_dictionary t ON act.tid = t.tid
WHERE
    t.target_chembl_id = 'ChEMBL205' -- Example target
    AND act.standard_type = 'IC50' -- Example activity type
    AND act.standard_value IS NOT NULL -- Ensure standard_value is not null
    AND act.standard_value::TEXT SIMILAR TO '[0-9.]+' -- Cast to text and use SIMILAR
TO
LIMIT 100;

```

-- Vietnamese

-- Lựa chọn các cột cần thiết từ các bảng khác nhau

```

SELECT
    act.activity_id, -- ID của hoạt động sinh học
    md.chembl_id, -- ID ChEMBL của phân tử
    cs.canonical_smiles, -- Cấu trúc SMILES của phân tử
    act.standard_type, -- Loại giá trị chuẩn (ví dụ: IC50)
    act.standard_value, -- Giá trị chuẩn
    act.standard_units, -- Đơn vị của giá trị chuẩn
    t.target_chembl_id -- ID ChEMBL của mục tiêu
FROM
    activities act -- Bảng hoạt động sinh học
JOIN
    molecule_dictionary md ON act.molregno = md.molregno -- Kết nối với bảng thông
    tin phân tử
JOIN
    compound_structures cs ON md.molregno = cs.molregno -- Kết nối với bảng cấu trúc
    phân tử
JOIN
    target_dictionary t ON act.tid = t.tid -- Kết nối với bảng thông tin mục tiêu
WHERE
    t.target_chembl_id = 'ChEMBL205' -- Lọc theo ID mục tiêu (ví dụ)
    AND act.standard_type = 'IC50' -- Lọc theo loại hoạt động (ví dụ)
    AND act.standard_value IS NOT NULL -- Đảm bảo giá trị chuẩn không rỗng
    AND act.standard_value::TEXT SIMILAR TO '[0-9.]+' -- Chuyển đổi sang text và sử
    dụng SIMILAR TO để so khớp
LIMIT 100; -- Giới hạn số lượng kết quả trả về là 100

```

- **Python Code Modifications:** The Python code for Examples 1, 2, or 3 can be used as a starting point, but the key is to keep the SQL query focused on a *specific target*.

Important Considerations:

- **Data Quality:** Always be mindful of data quality in ChEMBL. Check for outliers, inconsistencies in units, and potential errors.
- **Feature Engineering:** Experiment with different molecular descriptors and feature combinations. Feature engineering is often critical for building accurate models.

- **Model Selection:** Consider different machine learning models depending on your specific problem (e.g., Random Forests, Support Vector Machines).
- **Cross-Validation:** Use cross-validation to evaluate the performance of your models more robustly.

Remember to replace 'ChEMBL205' with the actual target ChEMBL ID you're interested in.

Let me know if you have a more specific research question related to Topic_ChEMBL_35_56, and I can provide more tailored code and guidance.