PharmApp Suite

🧠 AI for Drug Discovery and Development 🍼

🧩 Topic: Analyzing GPCR-targeted ligands

Okay, I understand. You're working with ChEMBL 35 data, using PostgreSQL and RDKit, within an AIMLops-structured project. You've encountered a couple of errors and need assistance with SQL queries and Python code for analysis, specifically focusing on filtering data to 100 rows for efficiency. You're also aiming to address the noted errors.

Let's break down the approach, providing solutions and examples in both English and Vietnamese.

1. Analysis Model and Overall Approach:

- **Goal:** To explore and analyze ChEMBL 35 data for drug discovery purposes, limited to 100 rows for manageable processing.
- **Data Source:** ChEMBL 35 database in PostgreSQL.
- **Tools:** PostgreSQL (psql, pgAdmin), RDKit, Jupyter Notebook, Python, potentially scikit-learn.
- **Analysis Focus:** While you haven't specified a specific analysis, I'll provide examples demonstrating common tasks like:
    - Filtering bioactivity data.
    - Calculating basic statistics.
    - Performing molecular property calculations with RDKit.
    - Correlation analysis.

2. Addressing the Errors:

- **Error a:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`
    - **Explanation:** This error arises because you're trying to use the regular expression operator `~` (similar to `LIKE` but for regular expressions) on a numeric column (`act.standard_value`). PostgreSQL likely requires you to cast this to text for regular expression matching. However, since you want to check if it's a number, a simpler approach is to use standard numeric comparison techniques.
    - **Solution:** Modify your SQL query to avoid regular expressions on the numeric column. Use comparison operators (`>`, `<`, `=`, etc.) or `IS NOT NULL`. If you *really* need to check for valid numeric input, consider a `CASE` statement or a stored procedure with error handling.
- **Error b:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function`
    - **Explanation:** Your scikit-learn version is outdated. The `squared=False` parameter, which returns the root mean squared error (RMSE) instead of the mean squared error (MSE), was introduced in a later version.
    - **Solution:** Update your scikit-learn installation: `pip install --upgrade scikit-learn`. If upgrading isn't possible, calculate the square root of the MSE manually using `numpy.sqrt(mean_squared_error(y_true, y_pred))`.

3. Code (SQL and Python) and Examples:

Let's assume your AIMLops project structure looks something like this:

```
my_chembl_project/
├── data/
```

```
│   └── chembl_activities.csv
├── notebooks/
│   └── Topic_CheMBL_35_22_1_data_extraction.ipynb
│   └── Topic_CheMBL_35_22_2_data_analysis.ipynb
├── scripts/
│   └── ...
└── ...
```

SQL (to extract data and save as CSV):

```sql
-- File: data/chembl_activities.sql

-- Extracting data from ChEMBL 35, filtering for specific activity type and limiting
to 100 rows.
-- This query assumes you're interested in IC50 values for a specific target.  Adjust
as needed.

\copy (
    SELECT
        act.activity_id,
        act.assay_id,
        act.standard_type,
        act.standard_relation,
        act.standard_value,
        act.standard_units,
        mol.molregno,
        md.chembl_id,
        md.pref_name
    FROM
        activities act
    JOIN
        assays ass ON act.assay_id = ass.assay_id
    JOIN
        target_dictionary td ON ass.tid = td.tid
    JOIN
        molecule_dictionary mol ON act.molregno = mol.molregno
    LEFT JOIN
        molecule_synonyms ms ON mol.molregno = ms.molregno
    LEFT JOIN
        compound_structures cs ON mol.molregno = cs.molregno
    LEFT JOIN
        compound_properties cp ON mol.molregno = cp.molregno
    LEFT JOIN
        molecule_hierarchy mh ON mol.molregno = mh.molregno
    LEFT JOIN
        molecule_atc ma ON mol.molregno = ma.molregno
    LEFT JOIN
        molecule_details md ON mol.molregno = md.molregno

    WHERE act.standard_type = 'IC50'
      AND act.standard_relation = '='
      AND act.standard_value IS NOT NULL -- Avoiding the regular expression issue
      AND act.standard_units = 'nM'
    LIMIT 100
) TO 'data/chembl_activities.csv' DELIMITER ',' CSV HEADER;
```

How to run this SQL:

1. Open pgAdmin.
2. Connect to your `chembl_35` database (192.168.206.136, rd/rd).

3. Open a new query window.
4. Paste the SQL code into the query window.
5. Execute the query. This will create the `chembl_activities.csv` file in your `data` directory.

Python (in Jupyter Notebook):

```
# File: notebooks/Topic_CheMBL_35_22_1_data_extraction.ipynb

import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Define the base path for your project
base_path = os.getcwd() # Get the current working directory (project root)
data_path = os.path.join(base_path, "data")
csv_file = os.path.join(data_path, "chembl_activities.csv")

# Load the data from the CSV file
try:
    df = pd.read_csv(csv_file)
    print(f"Data loaded successfully from {csv_file}")
    print(df.head()) # Display the first few rows
except FileNotFoundError:
    print(f"Error: The file {csv_file} was not found.  Ensure the SQL script was run
correctly.")
    df = None  # or handle the error as appropriate for your workflow


# Example 1: Basic Data Exploration (if df loaded successfully)
if df is not None:
    print("\nBasic Data Exploration:")
    print(f"Number of rows: {len(df)}")
    print(f"Number of columns: {len(df.columns)}")
    print(df.describe()) # Descriptive statistics

# Example 2: Add a column with the pIC50 value
    df['pIC50'] = -np.log10(df['standard_value'] * 1e-9) # Convert nM to M and
calculate -log10
    print("\nDataFrame with pIC50 column:")
    print(df.head())


    # Assuming you have a 'molregno' column and you want to add SMILES
    # This part requires looking up the SMILES from another table in ChEMBL or using a
pre-computed lookup.
    # For simplicity, I'm creating a dummy SMILES column.  You'll need to replace this
with actual data.
    # This is a placeholder to demonstrate RDKit integration.
    df['SMILES'] = 'CCO' # Replace with actual SMILES lookup
    print("Dataframe with SMILES column:")
    print(df.head())


#Example 3: RDKit integration to compute LogP
    # Function to calculate LogP using RDKit
    def calculate_logp(smiles):
        try:
```

```
            mol = Chem.MolFromSmiles(smiles)
            if mol is not None:
                logp = Descriptors.MolLogP(mol)
                return logp
            else:
                return None
        except:
            return None

    # Apply the function to the SMILES column
    df['LogP'] = df['SMILES'].apply(calculate_logp)
    print("Dataframe with calculated LogP:")
    print(df.head())

#Example 4: Analyze and visualize data

    import matplotlib.pyplot as plt
    import seaborn as sns

    # Basic distribution plot for pIC50 values
    plt.figure(figsize=(8, 6))
    sns.histplot(df['pIC50'].dropna(), kde=True) # Drop NaN values for plotting
    plt.title('Distribution of pIC50 Values')
    plt.xlabel('pIC50')
    plt.ylabel('Frequency')
    plt.show()

#Example 5: Correlation Analysis between LogP and pIC50 values
    correlation = df['LogP'].corr(df['pIC50'])
    print("Correlation between LogP and pIC50:", correlation)
```

Explanation of the Python Code:

1. **Import Libraries:** Imports necessary libraries (os, pandas, RDKit).
2. **Define Paths:** Sets up file paths using `os.path.join` for platform independence. This is crucial for AIMLops workflows.
3. **Load Data:** Reads the CSV file into a pandas DataFrame. Includes error handling for the case where the file isn't found.
4. **Data Exploration:** Prints the first few rows, the number of rows and columns, and descriptive statistics.
5. **pIC50 Calculation:** Calculates pIC50 values from IC50 values (converting from nM to M).
6. **SMILES Integration:** This is the *crucial* part for using RDKit. **You'll need to replace the placeholder `'CCO'` with actual SMILES strings.** This usually involves joining your `activities` data with a table containing molecule structures (e.g., `compound_structures`). I cannot write a concrete SQL for this part because I don't know your specific database schema and what columns contains SMILES or mol blocks.
7. **RDKit LogP Calculation:** Defines a function to calculate LogP using RDKit. Applies this function to the `SMILES` column to create a new `LogP` column.
8. **Data Visualization**: Use Matplotlib and Seaborn to plot the distribution of pIC50 values.
9. **Correlation Analysis**: Calculate the correlation between LogP and pIC50 values.

Vietnamese Translation:

1. Phân Tích Mô Hình và Tổng Quan Cách Tiếp Cận:

- **Mục tiêu:** Khám phá và phân tích dữ liệu ChEMBL 35 cho mục đích phát triển thuốc, giới hạn ở 100 dòng để xử lý dễ dàng.
- **Nguồn dữ liệu:** Cơ sở dữ liệu ChEMBL 35 trong PostgreSQL.

- **Công cụ:** PostgreSQL (psql, pgAdmin), RDKit, Jupyter Notebook, Python, có thể scikit-learn.
- **Trọng tâm phân tích:** Mặc dù bạn chưa chỉ định một phân tích cụ thể, tôi sẽ cung cấp các ví dụ minh họa các tác vụ phổ biến như:
    - Lọc dữ liệu hoạt tính sinh học.
    - Tính toán thống kê cơ bản.
    - Thực hiện tính toán thuộc tính phân tử bằng RDKit.
    - Phân tích tương quan.

2. Giải Quyết Các Lỗi:

- **Lỗi a:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`
    - **Giải thích:** Lỗi này phát sinh vì bạn đang cố gắng sử dụng toán tử biểu thức chính quy `~` (tương tự như `LIKE` nhưng dành cho biểu thức chính quy) trên một cột số (`act.standard_value`). PostgreSQL có thể yêu cầu bạn chuyển đổi cột này thành văn bản để so khớp biểu thức chính quy. Tuy nhiên, vì bạn muốn kiểm tra xem nó có phải là một số hay không, một cách tiếp cận đơn giản hơn là sử dụng các kỹ thuật so sánh số tiêu chuẩn.
    - **Giải pháp:** Sửa đổi truy vấn SQL của bạn để tránh các biểu thức chính quy trên cột số. Sử dụng các toán tử so sánh (>, <, =, v.v.) hoặc `IS NOT NULL`. Nếu bạn *thực sự* cần kiểm tra đầu vào số hợp lệ, hãy xem xét câu lệnh `CASE` hoặc một thủ tục lưu trữ với xử lý lỗi.
- **Lỗi b:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function`
    - **Giải thích:** Phiên bản scikit-learn của bạn đã cũ. Tham số `squared=False`, trả về căn bậc hai của lỗi bình phương trung bình (RMSE) thay vì lỗi bình phương trung bình (MSE), đã được giới thiệu trong một phiên bản sau.
    - **Giải pháp:** Nâng cấp cài đặt scikit-learn của bạn: `pip install --upgrade scikit-learn`. Nếu không thể nâng cấp, hãy tính căn bậc hai của MSE thủ công bằng cách sử dụng `numpy.sqrt(mean_squared_error(y_true, y_pred))`.

3. Mã (SQL và Python) và Ví Dụ:

Giả sử cấu trúc dự án AIMLops của bạn trông giống như sau:

```
my_chembl_project/
├── data/
│   └── chembl_activities.csv
├── notebooks/
│   └── Topic_CheMBL_35_22_1_data_extraction.ipynb
│   └── Topic_CheMBL_35_22_2_data_analysis.ipynb
├── scripts/
│   └── ...
└── ...
```

SQL (để trích xuất dữ liệu và lưu dưới dạng CSV):

```sql
-- Tệp: data/chembl_activities.sql

-- Trích xuất dữ liệu từ ChEMBL 35, lọc theo loại hoạt động cụ thể và giới hạn ở 100 hàng.
-- Truy vấn này giả định bạn quan tâm đến các giá trị IC50 cho một mục tiêu cụ thể. Điều chỉnh khi cần thiết.

\copy (
    SELECT
        act.activity_id,
        act.assay_id,
```

```
        act.standard_type,
        act.standard_relation,
        act.standard_value,
        act.standard_units,
        mol.molregno,
        md.chembl_id,
        md.pref_name
    FROM
        activities act
    JOIN
        assays ass ON act.assay_id = ass.assay_id
    JOIN
        target_dictionary td ON ass.tid = td.tid
    JOIN
        molecule_dictionary mol ON act.molregno = mol.molregno
    LEFT JOIN
        molecule_synonyms ms ON mol.molregno = ms.molregno
    LEFT JOIN
        compound_structures cs ON mol.molregno = cs.molregno
    LEFT JOIN
        compound_properties cp ON mol.molregno = cp.molregno
    LEFT JOIN
        molecule_hierarchy mh ON mol.molregno = mh.molregno
    LEFT JOIN
        molecule_atc ma ON mol.molregno = ma.molregno
    LEFT JOIN
        molecule_details md ON mol.molregno = md.molregno

    WHERE act.standard_type = 'IC50'
      AND act.standard_relation = '='
      AND act.standard_value IS NOT NULL -- Tránh vấn đề biểu thức chính quy
      AND act.standard_units = 'nM'
    LIMIT 100
) TO 'data/chembl_activities.csv' DELIMITER ',' CSV HEADER;
```

Cách chạy SQL này:

1. Mở pgAdmin.
2. Kết nối với cơ sở dữ liệu `chembl_35` của bạn (192.168.206.136, rd/rd).
3. Mở một cửa sổ truy vấn mới.
4. Dán mã SQL vào cửa sổ truy vấn.
5. Thực thi truy vấn. Điều này sẽ tạo tệp `chembl_activities.csv` trong thư mục `data` của bạn.

Python (trong Jupyter Notebook):

```
# Tệp: notebooks/Topic_CheMBL_35_22_1_data_extraction.ipynb

import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Xác định đường dẫn cơ sở cho dự án của bạn
base_path = os.getcwd() # Lấy thư mục làm việc hiện tại (gốc dự án)
data_path = os.path.join(base_path, "data")
csv_file = os.path.join(data_path, "chembl_activities.csv")

# Tải dữ liệu từ tệp CSV
try:
```

```
    df = pd.read_csv(csv_file)
    print(f"Dữ liệu đã được tải thành công từ {csv_file}")
    print(df.head()) # Hiển thị một vài hàng đầu tiên
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy tệp {csv_file}. Đảm bảo rằng tập lệnh SQL đã được chạy
chính xác.")
    df = None  # hoặc xử lý lỗi khi thích hợp cho quy trình làm việc của bạn

# Ví dụ 1: Khám phá dữ liệu cơ bản (nếu df được tải thành công)
if df is not None:
    print("\nKhám phá dữ liệu cơ bản:")
    print(f"Số lượng hàng: {len(df)}")
    print(f"Số lượng cột: {len(df.columns)}")
    print(df.describe()) # Thống kê mô tả

# Ví dụ 2: Thêm cột với giá trị pIC50
    df['pIC50'] = -np.log10(df['standard_value'] * 1e-9) # Chuyển đổi nM thành M và
tính -log10
    print("\nDataFrame với cột pIC50:")
    print(df.head())

    # Giả sử bạn có cột 'molregno' và bạn muốn thêm SMILES
    # Phần này yêu cầu tra cứu SMILES từ một bảng khác trong ChEMBL hoặc sử dụng tra
cứu được tính toán trước.
    # Để đơn giản, tôi đang tạo một cột SMILES giả. Bạn sẽ cần thay thế nó bằng dữ
liệu thực tế.
    # Đây là một trình giữ chỗ để minh họa tích hợp RDKit.
    df['SMILES'] = 'CCO'  # Thay thế bằng tra cứu SMILES thực tế
    print("Dataframe với cột SMILES:")
    print(df.head())

#Ví dụ 3: Tích hợp RDKit để tính toán LogP
    # Hàm để tính toán LogP bằng RDKit
    def calculate_logp(smiles):
        try:
            mol = Chem.MolFromSmiles(smiles)
            if mol is not None:
                logp = Descriptors.MolLogP(mol)
                return logp
            else:
                return None
        except:
            return None

    # Áp dụng hàm cho cột SMILES
    df['LogP'] = df['SMILES'].apply(calculate_logp)
    print("Dataframe với LogP đã tính toán:")
    print(df.head())

#Ví dụ 4: Phân tích và trực quan hóa dữ liệu
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Biểu đồ phân phối cơ bản cho các giá trị pIC50
    plt.figure(figsize=(8, 6))
    sns.histplot(df['pIC50'].dropna(), kde=True)  # Loại bỏ các giá trị NaN để vẽ đồ
thị
    plt.title('Phân phối của các giá trị pIC50')
    plt.xlabel('pIC50')
    plt.ylabel('Tần số')
```

```
    plt.show()

#Ví dụ 5: Phân tích tương quan giữa các giá trị LogP và pIC50
    correlation = df['LogP'].corr(df['pIC50'])
    print("Tương quan giữa LogP và pIC50:", correlation)
```

Giải thích Mã Python:

1. **Nhập thư viện:** Nhập các thư viện cần thiết (os, pandas, RDKit).
2. **Xác định đường dẫn:** Thiết lập đường dẫn tệp bằng `os.path.join` để độc lập với nền tảng. Điều này rất quan trọng đối với quy trình làm việc AIMLops.
3. **Tải dữ liệu:** Đọc tệp CSV vào DataFrame pandas. Bao gồm xử lý lỗi trong trường hợp không tìm thấy tệp.
4. **Khám phá dữ liệu:** In một vài hàng đầu tiên, số lượng hàng và cột và thống kê mô tả.
5. **Tính toán pIC50:** Tính toán các giá trị pIC50 từ các giá trị IC50 (chuyển đổi từ nM sang M).
6. **Tích hợp SMILES:** Đây là phần *quan trọng* để sử dụng RDKit. **Bạn sẽ cần thay thế trình giữ chỗ `'CCO'` bằng các chuỗi SMILES thực tế.** Điều này thường liên quan đến việc kết hợp dữ liệu `activities` của bạn với một bảng chứa cấu trúc phân tử (ví dụ: `compound_structures`). Tôi không thể viết SQL cụ thể cho phần này vì tôi không biết lược đồ cơ sở dữ liệu cụ thể của bạn và cột nào chứa SMILES hoặc mol block.
7. **Tính toán RDKit LogP:** Xác định một hàm để tính toán LogP bằng RDKit. Áp dụng hàm này cho cột `SMILES` để tạo một cột `LogP` mới.
8. **Trực quan hóa dữ liệu**: Sử dụng Matplotlib và Seaborn để vẽ biểu đồ phân phối các giá trị pIC50.
9. **Phân tích tương quan**: Tính toán tương quan giữa các giá trị LogP và pIC50.

Key Considerations and Next Steps:

- **SMILES Lookup:** The most important step is to correctly integrate SMILES data. Modify the SQL query to retrieve SMILES (or Mol blocks) associated with each `molregno`.
- **Error Handling:** The Python code includes basic `try...except` blocks. Expand this to handle more potential errors, such as invalid SMILES strings that cause RDKit to fail.
- **Data Cleaning:** Real-world ChEMBL data often contains missing values or inconsistencies. Implement data cleaning steps to handle these issues.
- **Feature Engineering:** Explore additional molecular descriptors and features that might be relevant to your analysis.
- **More Specific Analysis:** Define a more specific research question (e.g., “Can we predict IC50 values based on molecular properties for a specific target?”). This will guide your analysis and modeling efforts.
- **AIMLops Integration:** Consider using tools for experiment tracking (e.g., MLflow) to manage your models and experiments systematically.

This provides a solid foundation for your ChEMBL 35 analysis project. Remember to adapt the code and queries to your specific needs and research questions. Good luck!