# PostgreSQL QUERIES
# Exploring Targets Associated with Liver Diseases from ChEMBL Database

## 1 Objectives

The main objective of this second part of the project is to answer the following scientific questions using SQL queries in PostgreSQL (platform used: pgAdmin 4):

1. What are the target types and how many compounds have been tested against each target type associated with liver diseases in humans?
2. What is the distribution of activities for compounds targeting the top 5 most frequent targets associated with liver diseases, and how does this distribution vary by target type?

Initially, a new database has been created (e.g.,'target_liver') and used to store the processed data from the KNIME workflow.

## 2 SQL Queries

**QUESTION 1:**
**What are the target types and how many compounds have been tested against each target type associated with liver diseases in humans?**

The SQL query which answers question 1 is given in the following, alongside short comments for line:

```
CREATE VIEW liver_target_compound_count AS          -- create a new view
SELECT td.target_type,                              -- select colums
    COUNT(DISTINCT act.molregno) AS num_compounds   -- count of distinct compounds
FROM target_dictionary td                           -- source table
JOIN assays ass ON td.tid = ass.tid                 -- join tables
JOIN activities act ON ass.assay_id = act.assay_id
WHERE ass.description LIKE '%liver%'                -- filter results
GROUP BY td.target_type                             -- group results
ORDER BY num_compounds DESC;                        -- order results descending
```

This query creates a new view named *liver_target_compound_count* that retrieves information from the 'target_liver' database by joining three tables: **target_dictionary, assays**, and **activities**. The **assays** table is joined to the **target_dictionary** table using the common 'tid' (target ID) column, whereas the **activities** table is joined to the **assays** table using the common 'assay_id' column. To simplify their usage in queries, the three tables have been renamed as follows: **target_dictionary** to **td**, **assays** to **ass**, and **activities** to **act**. The query counts the number of distinct compounds for each 'target_type' in **ass** related to liver, based on the 'molregno' column which is a unique identifier for a given compound or molecule in the **act** table, and returns the

'target_type' and the count of distinct compounds as 'num_compounds'. The results are grouped by 'target_type' and ordered by the count of distinct compounds in descending order.

The view as shown here:

| | target_type<br>character varying | num_compounds<br>bigint |
|---|---|---|
| 1 | CELL-LINE | 68918 |
| 2 | SUBCELLULAR | 14705 |
| 3 | SINGLE PROTEIN | 11405 |
| 4 | TISSUE | 9463 |
| 5 | ORGANISM | 1348 |
| 6 | PROTEIN FAMILY | 317 |
| 7 | PROTEIN COMPLEX | 48 |
| 8 | PROTEIN-PROTEIN INTERACTION | 23 |
| 9 | PROTEIN COMPLEX GROUP | 2 |

can be useful for future queries that require this type of information and can also help identify target types that are frequently tested in assays related to liver.

**QUESTION 2:**

**What is the distribution of activities for compounds targeting the top 5 most frequent targets associated with liver diseases, and how does this distribution vary by target type?**

Before answering this question it is necessary to check for NULL entries in the 'standard_value' column which reports the measured activity values. This step is necessary because it is a common practice in assays to report the results of an experiment as 'not detected' or 'below the detection limit' when the measured signal is too low to accurately quantify the activity of the compound being tested. In such cases, the value of the 'standard_value' field is considered as a distinct category of results that cannot be treated as zero or any other numerical value and it is often set to NULL to indicate that no measurable activity was observed. Filtering for NULL values ensures that the analysis only considers the available data that can be meaningfully compared and interpreted. To check how many NULL entries are present (if any) in the 'standard_value' field of the **activities** table, the following query has been used:

```
SELECT COUNT(*) AS count
FROM activities
WHERE standard_value IS NULL;        -- filter NULL entries
```

which reveals 1 473 414 NULL entries in total:

| | count<br>bigint |
|---|---|
| 1 | 1473414 |

To see how this count of NULL entries is distributed across the different target types, the query from the first question can be modified to include additional aggregate functions COUNT() that counts the number of rows where 'standard_value' is NULL as follows:

```sql
SELECT td.target_type,
       COUNT(act.molregno) AS num_compounds,
       COUNT(act.standard_value) AS compounds_with_activity,
       COUNT(CASE WHEN act.standard_value IS NULL THEN 1 END) AS compounds_with_null_activity
FROM target_dictionary td
JOIN assays ass ON td.tid = ass.tid
JOIN activities act ON ass.assay_id = act.assay_id
WHERE ass.description LIKE '%liver%'
GROUP BY td.target_type
ORDER BY num_compounds DESC;
```

In this modified query, the COUNT(CASE WHEN act.standard_value IS NULL THEN 1 END) expression counts the number of rows where 'standard_value' is NULL for each 'target_type'. The result of this expression is included in the output as the compounds_with_null_activity column, whereas the compounds_with_activity column gives the count of rows where standard_value is not NULL. As before, the 'num_compounds' column gives the total number of compounds for each target type, regardless of whether they have an activity measurement or not. The output of this query is:

| | target_type character varying | num_compounds bigint | compounds_with_activity bigint | compounds_with_null_activity bigint |
|---|---|---|---|---|
| 1 | CELL-LINE | 69011 | 68967 | 44 |
| 2 | SINGLE PROTEIN | 26245 | 22700 | 3545 |
| 3 | SUBCELLULAR | 16924 | 14888 | 2036 |
| 4 | TISSUE | 12377 | 9909 | 2468 |
| 5 | ORGANISM | 2495 | 1440 | 1055 |
| 6 | PROTEIN FAMILY | 500 | 376 | 124 |
| 7 | PROTEIN-PROTEIN INTERACTION | 99 | 77 | 22 |
| 8 | PROTEIN COMPLEX | 49 | 49 | 0 |
| 9 | PROTEIN COMPLEX GROUP | 2 | 2 | 0 |

Focusing now on question 2, a subquery (inner query) is used to select the top 5 target types based on the number of unique molecules with activity data in assays that have the word "liver" in their description. This subquery retrieves the top 5 target types by joining and renaming the three tables **target_dictionary, assays**, and **activities**, in a similar manner as in question 1, and ordering the results based on the count of the unique molecules with activity data in descending order as shown here:

```sql
SELECT target_type                                  -- select columns
FROM target_dictionary td                           -- source table
JOIN assays ass ON td.tid = ass.tid                 -- join tables
JOIN activities act ON ass.assay_id = act.assay_id
WHERE ass.description LIKE '%liver%'                -- filter results
GROUP BY target_type                                -- group results
ORDER BY COUNT(DISTINCT act.molregno) DESC          -- order count descending
LIMIT 5;                                            -- limit: top 5 results
```

The output of this subquery is:

| | target_type character varying |
|---|---|
| 1 | CELL-LINE |
| 2 | SUBCELLULAR |
| 3 | SINGLE PROTEIN |
| 4 | TISSUE |
| 5 | ORGANISM |

In the main query, the SELECT statement specifies the columns to be returned in the result set. The query retrieves data from the **target_dictionary (td)**, **assays (ass)**, and **activities (act)** tables. The columns 'target_type' and 'pref_name' from the **td** table, 'num_compounds', 'min_activity', 'max_activity', 'avg_activity', and 'std_activity' are calculated by aggregate functions on the **act** table. The COALESCE() function is used to handle cases where the aggregate functions would return NULL values, such that these values are replaced by the zero 0 value, which can help avoid issues when performing further analyses or visualizations on the query output. The FROM clause lists the tables used in the query. The JOIN keyword is used to combine the tables based on their aforementioned relationship. The WHERE clause filters the rows in the result set based on specific condition. The query filters assays that have the string "liver" in their description column. It also filters on non-NULL entries in the 'standard_value' field of **act** and on **td** based on the subquery that selects the top 5 target types that have the highest number of distinct 'molregno' values in their activities table. The GROUP BY clause groups the result set by 'target_type' and 'pref_name'. The ORDER BY clause orders the result set in ascending order based on 'target_type' and in descending order based on the 'avg_activity'.

```
SELECT td.target_type, td.pref_name,
       COUNT(DISTINCT act.molregno) AS num_compounds,
       COALESCE(MIN(act.standard_value), 0) AS min_activity,    -- calc. minimum activity value
       COALESCE(MAX(act.standard_value), 0) AS max_activity,    -- calc. maximum activity value
       COALESCE(AVG(act.standard_value), 0) AS avg_activity,    -- calc. average activity value
       COALESCE(STDDEV(act.standard_value), 0) AS std_activity  -- calc. standard deviation
FROM target_dictionary td
JOIN assays ass ON td.tid = ass.tid
JOIN activities act ON ass.assay_id = act.assay_id
WHERE ass.description LIKE '%liver%'
  AND act.standard_value IS NOT NULL                -- filter NULL entries
  AND td.target_type IN (                           -- filter within top 5 targets
      SELECT target_type                            -- start of subquery
      FROM target_dictionary
      JOIN assays ON target_dictionary.tid = assays.tid
      JOIN activities ON assays.assay_id = activities.assay_id
      WHERE assays.description LIKE '%liver%'
      GROUP BY target_type
      ORDER BY COUNT(DISTINCT activities.molregno) DESC
      LIMIT 5
  )                                                 -- end of subquery
GROUP BY td.target_type, td.pref_name
ORDER BY td.target_type ASC, avg_activity DESC;
```

The first 15 rows of the output are shown here:

| | target_type character varying | pref_name character varying | num_compounds bigint | min_activity double precision | max_activity double precision | avg_activity double precision | std_activity double precision |
|---|---|---|---|---|---|---|---|
| 1 | CELL-LINE | SK-HEP1 | 7 | 2970 | 500560 | 90125.71428571429 | 181658.99398388754 |
| 2 | CELL-LINE | Huh-7 | 23 | 10000 | 100000 | 54312.5 | 30479.962234656505 |
| 3 | CELL-LINE | MCF7 | 2 | 3.981 | 90000 | 18230.7962 | 40120.984738097744 |
| 4 | CELL-LINE | HA22T cell line | 29 | 48 | 10000 | 3963.6206896551726 | 3199.733330560456 |
| 5 | CELL-LINE | BEL-7404 tumor cell line | 6 | 32 | 8370 | 2220.8333333333335 | 3235.70359684979 |
| 6 | CELL-LINE | Bel-7402 | 72 | 0.04 | 3980 | 818.2922972972974 | 1100.3544281724683 |
| 7 | CELL-LINE | MONO-MAC-6 | 1 | 800 | 800 | 800 | 0 |
| 8 | CELL-LINE | Cell line | 2 | 114 | 141 | 127.5 | 19.091883092036785 |
| 9 | CELL-LINE | COLO 205 | 1 | 76 | 76 | 76 | 0 |
| 10 | CELL-LINE | Hepatocyte | 43 | 0 | 99 | 26.968181818181822 | 33.70867511506029 |
| 11 | CELL-LINE | SNU-354 | 11 | 0.5 | 200 | 24.93583333333333 | 55.712553091468514 |
| 12 | CELL-LINE | HepG2 | 68704 | -218 | 200000 | 18.981241348205206 | 1780.4939750502676 |
| 13 | CELL-LINE | HeLa | 2 | 8 | 8 | 8 | 0 |
| 14 | CELL-LINE | Caov-3 cell line | 1 | 7 | 8 | 7.5 | 0.7071067811865476 |
| 15 | CELL-LINE | Caco-2 | 1 | 2.7 | 2.7 | 2.7 | 0 |