**Broken SQL Query Example (StackOverflow)**

The following SQL query (from a StackOverflow question) is intended to retrieve, for each country, its most populous city’s population and names. The query uses two CTEs (cte1, cte2) and a join. However, it contains errors and does not run correctly. The original (broken) query is shown below:

***WITH cte1 AS (***

***SELECT DISTINCT countrycode, name***

***FROM city***

***),***

***cte2 AS (***

***SELECT name, population***

***FROM city***

***WHERE cte2.countrycode = cte1.countrycode -- incorrect reference***

***ORDER BY population DESC***

***LIMIT 1***

***)***

***SELECT***

***population,***

***name,***

***name***

***FROM (***

***SELECT \****

***FROM cte1, cte2***

***) AS a;***

**Intended Query Purpose**

The user’s goal is to find the city with the largest population in each country, along with that city’s population. In other words, for each distinct countrycode, the query should return the highest-population city in that country, its population, and the country name. The query attempts to build two CTEs: one (cte1) listing each country, and another (cte2) selecting the top city by population per country, then join them to get one row per country.

**Problems in the Original Query**

**The original query fails due to several issues:**

* **Invalid Self-Reference in CTE:** Inside cte2, the WHERE clause uses cte2.countrycode and cte1.countrycode as if the CTEs were already available. You cannot reference a CTE from within its own definition (or from another CTE in that way). The parser reports an “unknown column” error for cte2.countrycode
* **Ambiguous Column Names:** The outer SELECT lists two columns both named name. After the join (or subquery), this produces duplicate column names, causing an error “duplicate column name 'name'”.Each selected column must have a unique alias if names repeat.
* **Improper Subquery Join:** The query does FROM (SELECT \* FROM cte1, cte2) AS a without a valid join condition at the subquery level. This treats cte1 and cte2 as cartesian-joined (though it then filters by the WHERE in cte2), which is both inefficient and semantically unclear. In particular, the \G at the end (MySQL client row mode) is not valid SQL and should be removed.
* **Missing CTE Correlation:** The intent was to correlate cte1 and cte2 by country, but the attempt to do so via a subquery failed. Instead, a JOIN should be used to relate the two CTEs after they are fully defined.

**Corrected Query**

Below is a corrected version using a window function (ROW\_NUMBER()) to pick the top city per country. This approach properly separates the CTE definitions and then joins them:

***WITH cte1 AS (***

***SELECT DISTINCT countrycode, name***

***FROM city***

***),***

***cte2 AS (***

***SELECT***

***name,***

***population,***

***countrycode,***

***ROW\_NUMBER() OVER (***

***PARTITION BY countrycode***

***ORDER BY population DESC***

***) AS Rn***

***FROM city***

***)***

***SELECT***

***cte2.population,***

***cte2.name AS CityName,***

***cte1.name AS CountryName***

***FROM cte1***

***JOIN cte2***

***ON cte2.countrycode = cte1.countrycode***

***WHERE cte2.Rn = 1;***

**Explanation of Fixes**

* **Separate CTE Logic and Use of Window Function:** We redefined cte2 to select each city with a row number (ROW\_NUMBER()) partitioned by countrycode, ordered by population descending. This ranks cities within each country so that Rn = 1 picks the largest city per country.
* **Proper JOIN Instead of Subquery Trick:** Instead of combining cte1 and cte2 via a derived table, we perform a direct JOIN on countrycode. This cleanly relates the country list (cte1) to the ranked cities (cte2).
* **Unique Column Names:** We alias output columns to avoid ambiguity: cte2.name AS CityName and cte1.name AS CountryName. This ensures the two name columns are distinct in the final result.
* **Removal of Illegal References:** The corrected query no longer tries to reference cte1 or cte2 inside the other CTE definition. Each CTE is self-contained. The join condition is moved to the final JOIN ... ON clause, which is the correct way to correlate the two sets.
* **Removal of \G:** The MySQL client \G terminator is removed as it is not valid in a standard SQL script.

These changes fix the syntax errors and implement the intended logic (returning each country’s most populous city and population) correctly.