**Report: Inconsistency in Gene/Biopolymer Queries**

**Scenario**

In the current implementation, gene/biopolymer queries are producing inconsistent results across multiple executions due to a mismatch between source\_id values in the database. Specifically:

1. In the first execution, genes/biopolymers with source\_id = 0 are excluded from the results.
2. When rerunning the query with genes that were excluded initially, these genes incorrectly appear in the output, now associated with valid source\_id values (e.g., 3, 5, 7), even though such mappings do not exist in the db database.

**Root Cause Analysis**

**Problem Source**

* **Invalid source\_id = 0:** Genes/biopolymers in the group\_biopolymer table (d\_gb) have source\_id = 0, which represents either an undefined or placeholder value. However, the query does not appropriately filter these records or ensure they are handled separately.
* **Faulty Joins or Filters:** The query combines source\_id values across temporary tables (m\_bg and m\_c) and the main database (db). This allows invalid mappings between genes with source\_id = 0 and valid source\_id values (3, 5, 7), leading to inconsistent outputs.

**Impact**

1. Incorrect results: Genes with source\_id = 0 appear as if they are associated with valid sources in subsequent executions.
2. Ambiguity: It becomes unclear which results are valid, leading to potential misinterpretations in downstream analyses.

**Proposed Solutions**

**1. Quick Fix: Exclude source\_id = 0**

This solution ensures that genes with source\_id = 0 are excluded entirely from the query. While this resolves the immediate inconsistency, it may not address the underlying issue of source\_id = 0 entries in the database.

**SQL Implementation:**

*SELECT*

*m\_bg.label AS gene\_label,*

*d\_g.label AS group\_label,*

*m\_c.label AS source\_label,*

*(COALESCE(m\_bg.biopolymer\_id,'')||'\_'||COALESCE(d\_g.group\_id,'')||'\_'||COALESCE(m\_c.source\_id,'')) AS \_rowid*

*FROM*

*`db`.`group` AS d\_g,*

*`db`.`group\_biopolymer` AS d\_gb,*

*main`.`gene` AS m\_bg,*

*`main`.`source` AS m\_c*

*WHERE*

*(d\_gb.specificity >= 100 OR d\_gb.specificity >= 100) AND*

*d\_g.group\_id = d\_gb.group\_id AND*

*d\_g.source\_id = m\_c.source\_id AND*

*d\_gb.biopolymer\_id != 0 AND*

*d\_bg.source\_id != 0 AND*

*d\_g.source\_id != 0 AND*

*d\_gb.group\_id = d\_g.group\_id AND*

*m\_bg.biopolymer\_id = d\_gb.biopolymer\_id AND*

*m\_c.source\_id = d\_g.source\_id*

**Advantages:**

* Immediate resolution to the inconsistency.
* Minimal changes required, quick to implement.

**Disadvantages:**

* Genes with source\_id = 0 are completely excluded, which may result in data loss if those records are relevant.
* Does not address the root cause of the invalid source\_id = 0.

**2. Best Practice: Use Explicit Joins**

This approach rewrites the query using **explicit joins** to ensure clarity and proper handling of source\_id = 0. It involves explicitly defining relationships between tables, ensuring that invalid mappings are avoided.

**SQL Implementation:**

*SELECT*

*m\_bg.label AS gene\_label,*

*d\_g.label AS group\_label,*

*m\_c.label AS source\_label,*

*(COALESCE(m\_bg.biopolymer\_id, '') || '\_' || COALESCE(d\_g.group\_id, '') || '\_' || COALESCE(m\_c.source\_id, '')) AS \_rowid*

*FROM*

*`db`.`group\_biopolymer` AS d\_gb*

*INNER JOIN `db`.`group` AS d\_g*

*ON d\_gb.group\_id = d\_g.group\_id*

*INNER JOIN `m\_bg` AS m\_bg*

*ON d\_gb.biopolymer\_id = m\_bg.biopolymer\_id*

*LEFT JOIN `m\_c` AS m\_c*

*ON d\_gb.source\_id = m\_c.source\_id AND d\_gb.source\_id != 0 -- Ensure proper source filtering*

*WHERE*

*d\_gb.specificity >= 100 AND*

*d\_gb.biopolymer\_id != 0;*

**Advantages:**

* Ensures clear relationships between tables and avoids invalid mappings.
* Retains records with source\_id = 0, allowing for separate handling or further analysis if needed.
* Follows SQL best practices, improving long-term maintainability.

**Disadvantages:**

* Requires more significant changes to the query structure.
* May take longer to implement and test due to the added complexity.

**Recommendations**

1. **Immediate Resolution:**
   * Implement the **Quick Fix** solution to resolve the inconsistency immediately. This ensures valid outputs while giving time to address the root cause.
2. **Long-Term Improvement:**
   * Plan and implement the **Best Practice** solution to ensure proper handling of source\_id = 0 and to align the query with SQL best practices.
3. **Future Considerations:**
   * Investigate the root cause of source\_id = 0 in the database. If it represents missing or placeholder data, consider strategies for proper handling, such as mapping to valid sources or introducing a "null source" category.

**Summary Table**

| **Solution** | **Time to Implement** | **Resolves Issue** | **Maintains Data Integrity** | **Long-Term Maintainability** |
| --- | --- | --- | --- | --- |
| Quick Fix | Low | Yes | No | No |
| Best Practice | Medium | Yes | Yes | Yes |