**ETL Data Discrepancy Analysis: ENTMOD vs POAIND Tables**

**Problem Analysis**

Based on the provided screenshots, I can see you're performing a **diff analysis** between two Oracle tables:

* **ENTMOD\_WEEKLY\_POST\_BACKUP\_06292025** (legacy/backup table)
* **ENTMOD\_WEEKLY\_POST\_SNAPSHOT\_06292025** (new ETL-processed table)

The analysis involves comparing records using a comprehensive MINUS operation on all fields to identify discrepancies.

**Key Observations from the Code**

**1. Comparison Strategy**

* Using SELECT \* MINUS SELECT \* approach to find differences
* Comparing identical field sets between two tables with date suffix (06292025)
* Extensive field list (60+ columns) including financial, demographic, and metadata fields

**2. Java ETL MERGE Operations**

From the MERGE statements visible in images 2-5, I can see:

**Critical Issue Identified**: The MERGE logic shows **inconsistent join conditions** between different steps:

* **Step 1 (Image 2)**: Uses EMQOSID, PERIOD, MFT, MFTCD, TYPE as matching criteria
* **Step 2 (Image 3)**: Uses EMQOSID, PERIOD, FDTAXPRD, MFT, FTQEJID, TYPEID
* **Step 3 (Image 4)**: Uses EMQOSID, PERIOD, FDTAXPRD, MFT, FTMFTCD, TYPEID

This **inconsistency in join logic** is likely the primary source of your data discrepancies.

**Root Cause Analysis**

**Primary Issue: Inconsistent MERGE Logic**

1. **Different Join Keys Across Steps**
   * Some merges use MFT vs MFTCD vs FTMFTCD
   * Some include FDTAXPRD, others don't
   * TYPE vs TYPEID inconsistency
   * Missing FTQEJID in some merges
2. **Business Logic Implications**
   * Records that should match may not be matching due to different join criteria
   * This could result in duplicates, missing updates, or incorrect inserts
   * Different processing steps may be operating on different logical entities

**Secondary Issues**

1. **Data Type/Format Differences**
   * Date fields: Multiple date formats visible ('01/01/1900', 'MM/DD/YYYY')
   * Status fields: String vs numeric representations ('0' vs 0)
2. **Null Handling Variations**
   * Some MERGE operations use IS NOT NULL conditions
   * Different null substitution strategies across steps
3. **Field Mapping Inconsistencies**
   * Some fields populated differently across MERGE steps
   * Potential transformation logic differences

**Specific Resolution Steps**

**Immediate Actions**

1. **Standardize MERGE Join Conditions**
2. -- Audit your Java ETL code to ensure consistent join logic
3. -- All MERGE operations should use the same business key definition
4. -- Example standardized join:
5. ON (a.EMQOSID = b.EMQOSID
6. AND a.PERIOD = b.PERIOD
7. AND a.FDTAXPRD = b.FDTAXPRD
8. AND a.MFT = b.MFT
9. AND a.TYPEID = b.TYPEID)
10. **Field Mapping Verification**
11. -- Check for specific field differences
12. SELECT EMQOSID, MFT, MFTCD, FTMFTCD, TYPE, TYPEID, FTQEJID
13. FROM ENTMOD\_WEEKLY\_POST\_BACKUP\_06292025
14. MINUS
15. SELECT EMQOSID, MFT, MFTCD, FTMFTCD, TYPE, TYPEID, FTQEJID
16. FROM ENTMOD\_WEEKLY\_POST\_SNAPSHOT\_06292025;

**Detailed Troubleshooting**

1. **Isolate Join Key Issues**
2. -- Check if records exist with same business meaning but different technical keys
3. SELECT b.EMQOSID, b.PERIOD, b.MFT, b.MFTCD, b.FTMFTCD, b.TYPE, b.TYPEID
4. FROM ENTMOD\_WEEKLY\_POST\_BACKUP\_06292025 b
5. WHERE NOT EXISTS (
6. SELECT 1 FROM ENTMOD\_WEEKLY\_POST\_SNAPSHOT\_06292025 s
7. WHERE b.EMQOSID = s.EMQOSID
8. AND b.PERIOD = s.PERIOD
9. );
10. **Validate Transformation Logic**
    * Review your Java ETL code for each mergeIntoEntmodStep
    * Ensure field transformations are applied consistently
    * Verify date format handling and null value substitutions
11. **Step-by-Step Data Validation**
12. -- Create intermediate validation tables after each merge step
13. CREATE TABLE ENTMOD\_AFTER\_STEP1 AS SELECT \* FROM ENTMOD;
14. -- Run your merge step 1
15. CREATE TABLE ENTMOD\_AFTER\_STEP2 AS SELECT \* FROM ENTMOD;
16. -- Compare differences between each step

**Java ETL Code Review Focus Areas**

1. **Configuration Management**
   * Check if different environments use different field mappings
   * Verify source table column definitions match expectations
   * Ensure all required fields are being selected from source
2. **Transaction Boundaries**
   * Verify each MERGE operation commits successfully
   * Check for transaction isolation issues
   * Ensure proper error handling doesn't skip records

**Expected Outcome**

After implementing these fixes, your MINUS query should return zero rows, indicating complete data consistency between the backup and ETL-processed tables.

**Next Steps**

1. **Immediate**: Fix the inconsistent join conditions in your Java MERGE operations
2. **Short-term**: Implement the field-by-field comparison queries to identify specific discrepancies
3. **Long-term**: Add data validation checkpoints after each ETL step to prevent future discrepancies

Would you like me to help you dive deeper into any specific aspect of this analysis or provide more detailed SQL for isolating particular types of discrepancies?