STOP USING .WITHCOLUMN() IN A LOOP

FROM 20 CALLS TO 1

```python
# The anti-pattern
for col_name, expr in mapping.items():
df = df.withColumn(col_name, expr)

# The fix
df = df.withColumns(mapping)
```

THE PROBLEM

Sign Up to the newsletter

Databricks.news

Find more databricks tips @

dailydatabricks.tips

WHY .WITHCOLUMN() IN A LOOP HURTS

- **New DataFrame Every Call** — Each .withColumn() returns a brand-new DataFrame object with a fresh Project node on the logical plan.
- **Nested Plan Explosion** — 50 loop iterations = 50 nested Project nodes that Catalyst has to traverse and optimize through.
- **StackOverflowError Risk** — With hundreds of columns (common in wide fact tables or IoT data), recursive plan traversal can crash.
- **Planning Time Bloat** — Even before crashing, analysis and optimization passes scale with plan depth, turning milliseconds into seconds.

THE FIX

.WITHCOLUMNS() — SPARK 3.3+

Pass a dictionary of column expressions. One call, one plan node, same result.

```python
# One dict, one plan node
df_result = df.withColumns({
"date_upper": F.upper(col("trnx_date")),
"name_clean": F.trim(col("custName")),
"bal_abs": F.abs(col("`ACCT-bal`")),
"amt_rounded": F.round(col("trnxAmt"), 2),
"loaded_at": F.current_timestamp(),
})
```

REAL WORLD USE

🏷️

Metadata Enrichment

Add ingestion_timestamp, source_system, row_hash in one pass

🔢

Type Casting

Cast and format multiple columns without plan bloat

🧹

Column Standardisation

snake_case + abbreviation expansion across all columns

📋

Business Rules

Apply derived column logic from config or lookup tables

🔐

Data Validation

Add is_valid, row_quality flags in a single step

⚡

Audit Columns

pipeline_run_id, load_date, environment tags at once

Sign Up to the newsletter

Databricks.news

Find more databricks tips @

dailydatabricks.tips

BONUS: RENAME

.WITHCOLUMNSRENAMED() — SPARK 3.4+

Same anti-pattern, same fix. Dict of old name to new name, one plan node.

```python
# Same problem, same fix
df.withColumnsRenamed({
"CustID": "customer_id",
"trnxAmt": "transaction_amount",
"ACCT-bal": "account_balance",
"pmt$Status": "payment_status",
"addr.Line1": "address_line_1",
})
```

Build the dict dynamically with regex for snake_case + abbreviation expansion.

Sign Up to the newsletter

Databricks.news

Find more databricks tips @

dailydatabricks.tips

WANT MORE TIPS?

FOLLOW FOR DAILY DATABRICKS TIPS

SIGN UP TO THE NEWSLETTER

Databricks.news

dailydatabricks.tips