Lab 1 — Basic Transformation: Cleaning Nulls & Type Casting (Beginner)

Trainer note: This updated version reflects the exact working code you provided. No code has been changed; the lab explanation now aligns perfectly with the code cells you will run in Databricks.

Learning Objective

This lab introduces beginners to basic data transformation in Databricks. Learners will load data from a CSV file, inspect its structure, identify null values, remove invalid rows, clean numeric fields, convert data types, handle date parsing using `try_to_date`, remove duplicates, and finally write the transformed data into the silver layer as a Delta table.

Learning Outcomes

By the end of this lab, learners will be able to: - Read raw CSV data into a Spark DataFrame. - Explore schema and identify data quality issues. - Count null or blank values across columns. - Remove rows with missing mandatory fields. - Cast columns to proper data types. - Clean formatted numeric string values. - Parse dates using `try_to_date`. - Drop duplicates. - Save cleaned data in Delta format.

Dataset

You should upload your `customer_raw.csv` file into the following location:

```
/Volumes/workspace/default/test/customer_raw.csv
```

This is the path used throughout the code.

Step-by-Step Lab Instructions

Run each cell one by one and observe the output. Pause after each operation to discuss what is happening.

Step 1 — Read the CSV File

The dataset is loaded without inferring the schema. All columns will initially be treated as strings.

```
# Replace path if needed
csv_path = '/Volumes/workspace/default/test/customer_raw.csv'

# Read CSV without forcing schema (we'll inspect and then convert types)
df_raw = spark.read.option('header', 'true').option('inferSchema',
'false').csv(csv_path)

display(df_raw.limit(20))
```

Explain to learners why `inferSchema=false` is intentional: beginners must see the raw incoming types.

Step 2 — Inspect Schema and Sample Rows

```
df_raw.printSchema()
df_raw.show(10, truncate=False)
```

Use this moment to highlight that all fields appear as strings.

Step 3 — Count Null and Blank Values

This step identifies data quality issues column by column.

```
from pyspark.sql import functions as F

null_counts = df_raw.select([
    F.count(F.when(F.col(c).isNull() | (F.col(c) == ''), c)).alias(c)
    for c in df_raw.columns
])

display(null_counts)
```

Train learners to understand both `null` and empty-string cases.

Step 4 — Remove Rows Missing Critical Fields

Rows missing `customer_id` or `email` are removed.

```
# Define a cleaned DataFrame by dropping rows where customer_id or email is
null/blank
cleaned = (
    df_raw.filter((F.col('customer_id').isNotNull()) & (F.col('customer_id') !=
''))
          .filter((F.col('email').isNotNull()) & (F.col('email') != ''))
```

```
)

# Verify counts
print('Raw count:', df_raw.count())
print('After dropping critical nulls:', cleaned.count())
```

Discuss why these two fields are mandatory.

Step 5 — Clean Numeric Fields and Cast Column Types

Formatting issues like commas in numbers are fixed, and data types are converted.

```
from pyspark.sql.functions import regexp_replace, try_to_date

cleaned = cleaned.withColumn('customer_id', F.col('customer_id').cast('int'))

cleaned = cleaned.withColumn('total_spend_clean',
regexp_replace(F.col('total_spend'), ',', ''))
cleaned = cleaned.withColumn(
    'total_spend',
    F.when(F.col('total_spend_clean') == '', None)
     .otherwise(F.col('total_spend_clean').cast('double'))
)
cleaned = cleaned.drop('total_spend_clean')
```

Explain why financial numbers often arrive as formatted strings.

Step 6 — Parse Dates Using `try_to_date`

This gracefully handles invalid or inconsistent date formats.

```
cleaned = cleaned.withColumn(
    'signup_date',
    try_to_date(F.col('signup_date'), 'yyyy-MM-dd')
)

cleaned.printSchema()
display(cleaned.limit(20))
```

Discuss how `try_to_date` differs from `to_date`.

Step 7 — Remove Duplicate Rows

```
cleaned = cleaned.dropDuplicates()
```

Introduce full-row deduplication and mention that advanced strategies will come later.

Step 8 — Quick Summary and Basic Exploration

```
print('Cleaned count:', cleaned.count())
cleaned.describe(['total_spend']).show()
```

Review total counts and distribution of numeric columns.

Step 9 — Write Cleaned Data to Silver Layer

The data is saved as a Delta table.

```
silver_path = '/Volumes/workspace/default/test/customer_silver'

cleaned.write.format('delta').mode('overwrite').option('overwriteSchema',
'true').save(silver_path)
```

Explain the importance of Delta Lake for ACID guarantees and future transformations.

Trainer Closing Notes

This lab completes the first essential transformation workflow: reading raw CSV, cleaning critical fields, performing type conversions, parsing dates, eliminating duplicates, and writing a clean Delta dataset. The next lab will build on this by introducing more nuanced transformations, handling multiple date formats, and implementing data enrichment.

End of Updated Lab 1 — Basic Transformation