Polars: Handling Missing Values

A Comprehensive Guide

Introduction

This guide covers comprehensive techniques for handling missing values in Polars, a fast DataFrame library for Python. Missing values are a common challenge in data analysis, and Polars provides powerful tools to detect, remove, and fill them effectively.

1. Identifying Missing Values

Understanding null vs NaN:

Feature	null	NaN
Origin	Missing information	Invalid math operation
Python	None	np.nan
Detection	is_null()	is_nan()
Counting	null_count()	is_nan().sum()

Basic Detection:

```
import polars as pl

# Count nulls in all columns
df.null_count()

# Count nulls in specific column
df.select(pl.col('column').null_count())

# Count NaN values
df.select(pl.col('column').is_nan().sum())

# Filter rows with null
df.filter(pl.col('column').is_null())
```

2. Deleting Missing Values

Sometimes the best approach is to remove rows or columns with missing values. However, this should be done carefully to avoid losing important data.

```
# Drop rows with any null
df.drop_nulls()

# Drop rows with null in specific columns
df.drop_nulls(subset=['col1', 'col2'])

# Drop null AND NaN
df.with_columns(pl.all().fill_nan(None)).drop_nulls()

# Drop a column
df.drop('column_name')
```

3. Filling Missing Values

Filling Strategies:

Method	Use Case	Example
fill_null(0)	Default values	Rainfall, counts
fill_null(strategy="mean")	Statistical data	Surveys, measurements
interpolate()	Time series	Temperatures, prices
forward_fill()	Constant until change	Stock prices, inventory
backward_fill()	Scheduled events	Future predictions

```
# Fill with constant value
pl.col('column').fill_null(0)

# Fill with statistical measures
pl.col('column').fill_null(strategy='mean')
pl.col('column').fill_null(strategy='min')
pl.col('column').fill_null(strategy='max')

# Fill with custom expression
pl.col('column').fill_null(pl.col('column').median())

# Interpolation
pl.col('column').interpolate()
pl.col('column').interpolate(method='nearest')

# Forward/Backward fill
pl.col('column').forward_fill()
pl.col('column').forward_fill(limit=2)
pl.col('column').backward_fill()
```

Best Practices

1. Always inspect your data before deciding how to handle missing values
2. Convert NaN to null before using drop_nulls() - they are different!
3. Choose filling strategy based on data context and business logic
4. Document your decisions about missing value handling
5. Test the impact of your strategy on downstream analysis
6. Consider the percentage of missing values before deletion
7. Use with_columns() to preserve all columns when filling
8. Sort data before using interpolation for time series

Quick Reference

```
# Detection
df.null_count() # Count nulls
df.select(pl.col('x').is_nan().sum()) # Count NaN
df.filter(pl.col('x').is_null()) # Rows with null

# Deletion
df.drop_nulls() # Drop rows with null
df.drop('col') # Drop column

# Filling - Constant
pl.col('x').fill_null(0) # Fill with zero
pl.col('x').fill_nan(None) # Convert NaN to null

# Filling - Statistical
pl.col('x').fill_null(strategy='mean') # Mean
pl.col('x').fill_null(strategy='min') # Minimum
pl.col('x').fill_null(strategy='max') # Maximum

# Filling - Advanced
pl.col('x').interpolate() # Interpolation
pl.col('x').forward_fill() # Forward fill
pl.col('x').backward_fill() # Backward fill
```

For more information:
Visit the official Polars documentation at https://pola-rs.github.io/polars/