Handling Nulls & Missing Data in PySpark

Handling Nulls & Missing Data in PySpark

Nulls and missing values are common in real-world datasets. PySpark provides several powerful functions to **handle, clean, and process null values** efficiently.

1. Detecting Nulls

You can detect null values using **isNull()** and **isNotNull()** functions.

```
from pyspark.sql.functions import col #

Filter rows where "age" is null
df.filter(col("age").isNull()).show()
# Filter rows where "age" is not null
df.filter(col("age").isNotNull()).show()
```

2. Dropping Null Values — dropna()

Removes rows with nulls.

```
# Drop rows with ANY nulls
df.na.drop().show()

# Drop rows if ALL columns are null
df.na.drop(how="all").show()
```

```
# Drop rows only if specific column(s) have nulls
df.na.drop(subset=["name", "city"]).show()
```

- **`how='any'`(default)** → drop if any column is null
- **`how='all'`** → drop only if all specified columns are null

◆ 3. Replacing Nulls — `fillna() / na.fill()`

Fill null values with constants (per column or all).

```
# Fill all numeric nulls with 0
df.fillna(0).show()

# Fill string column nulls
df.fillna({'name': 'Unknown', 'city': 'NA'}).show()
# Equivalent using na.fill()
df.na.fill({'name': 'Unknown', 'city': 'NA'}).show()
```

◆ 4. Replacing Specific Values — `replace()`

Replaces specific values — including nulls if explicitly given.

```
# Replace None or NaN values
df.replace(to_replace=None, value="Unknown").show()

# Replace multiple values
df.replace(["", None], "NA", subset=["name"]).show()
```

Use `replace()` to transform values ***before*** or ***after*** null handling.

5. Conditional Null Handling — `when & otherwise`

You can apply logic to replace nulls based on conditions.

```
from pyspark.sql.functions import when

df.withColumn("city",

    when(col("city").isNull(), "NoCity")
    .otherwise(col("city"))
).show()
```

This is great when filling nulls based on **custom rules**.

6. Counting Nulls Per Column

Check how many nulls each column contains.

```
from pyspark.sql.functions import sum, col, isnan

df.select([

    sum(col(c).isNull().cast("int")).alias(c)
    for c in df.columns
]).show()
```

For NaN values (like float), add isnan() check:

```
df.select([
    sum((col(c).isNull() | isnan(c)).cast("int")).alias(c)
    for c in df.columns
]).show()
```

🔹 7. Filtering Based on Multiple Null Conditions

```
df.filter(
    (col("age").isNull()) & (col("city").isNull())
).show()
```

Combine conditions using &, |, ~ (NOT).

🔹 8. Filling Nulls with Previous/Next Values (Windowing)

Use **window functions** for forward or backward fill (like in Pandas).

```
from pyspark.sql.window import Window
from pyspark.sql.functions import last

windowSpec = Window.partitionBy("id").orderBy("date").rowsBetween(-
sys.maxsize, 0)
# Forward fill nulls
df.withColumn("amount_ffill", last("amount",
ignorenulls=True).over(windowSpec)).show()
```

🔹 9. Handling Nulls in Aggregation

Nulls are **ignored by default** in most aggregation functions:

```
# Null-safe average
df.groupBy("department").agg({"salary": "avg"}).show()
```

But you can use coalesce() to replace nulls before aggregating:

```
from pyspark.sql.functions import coalesce

df.withColumn("salary", coalesce("salary", lit(0))) \
  .groupBy("department") \
  .sum("salary") \
  .show()
```

🔹 10. Null-Safe Comparison — <=> Operator

Use `eqNullSafe()`or SQL `<=>`for **null-safe equals** (like SQL IS NOT DISTINCT FROM):

```
from pyspark.sql.functions import expr

df.filter(expr("name <=> 'John'")).show()
```

== fails when comparing with null

`<=>` works even if one side is null

🔹 11. Using SQL to Handle Nulls

Register table and run SQL:

```
df.createOrReplaceTempView("people")
```

```
spark.sql("""
SELECT *,
       CASE WHEN city IS NULL THEN 'Unknown' ELSE city END AS
clean_city
FROM  people
""").show()
```

✅ Summary: What to Use When?

Task	Function
Drop nulls	`dro pna ()`
Fill missing values	`fillna() / na.fill()`
Replace specific values	`rep lac e( )`
Conditional replacement	`when / otherwise`
Detect nulls	`isNull() /isNotNull()`
Null-safe equality check	`<=> / eqNullSafe()`
Window fill (forward/backward)	last() overwindow
Count nulls	`sum(isNull().cast())`
SQL-based handling	`CASE WHEN ...`

Seekho Bigdata

Data is the New Oil

Let’s build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/