fill() and fillna()

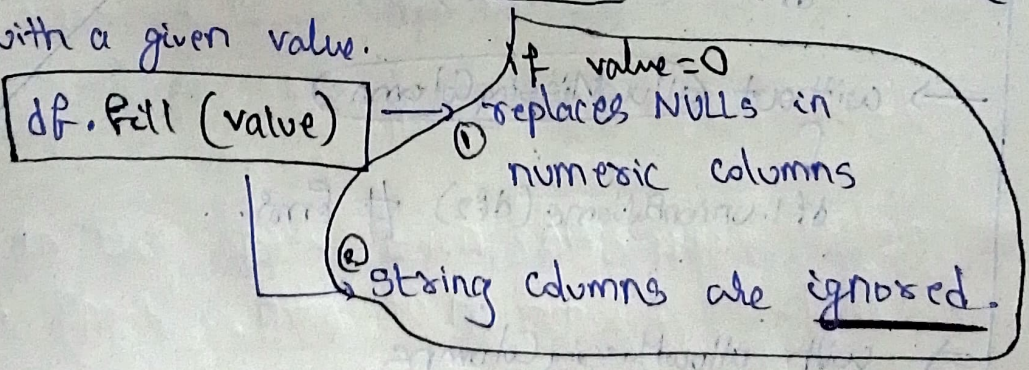
⇒ Those are used to handle the NULL values

List of NULL-handling functions:

1. fill()
2. fillna()
3. dropna()
4. isNull() / isNotNull()
5. coalesce()
6. when(), otherwise()
7. na.replace()

fill()

- used to replace NULL values in a DataFrame with a given value.



⇒ df.fill(value)

① ⇒ df.fill(5)

- replaces NULLs in numeric columns
- string columns are ignored

⇒ ② df.fill(5, subset=["col1", "col2"])

: only these columns are affected

③ Fill different values for different columns.

```
df.fill(
        {
            "col1" : "val1",
            "col2" : 5,
            "col3" : "unknown",
            "col4" : "NA"
        }
)
```

- Column-wise replacement
- very common in real projects

④ Data Type rules

Column Type	Fill value allowed
Integer	int
Double	int / double
String	String
Date	X (not directly)

Ex
```
df.fill("NA", subset=["units sold"])
```
(int)

fillna():

- replaces NULL values in a Dataframe with a specified value.

> **Note:**
> - fillna() & fill() are exactly the same
> - There is no behaviour difference

```
df.fill(5)
df.fillna(5, subset = ['col1', 'col2'])
df.fillna( {
        'col1' : 5,
        'col2' : 'val1',
        'col3' : 'Unknown'
    } )
```

→ just same as fill()

Why fill() & fillna() if both do same job?

- In early spark version, APIs were evolving
- Spark wanted to be familiar to Pandas users

> Pandas has : df.fillna()
>
> Spark introducted the same : df.fillna()

Therefore;

> Later, spark also exposed : df.fill()
>
> as a ghost alias

dropna()

- removes rows that contains NULL values
- unlike fills()/fillna() which replaces NULLS, dropna() deletes rows.

⇒ 1) `df.dropna()`

- If atleast one column is NULL → that specific row is dropped.
- same as : `df.dropna(how="any")`

2) `df.dropna(how="all")`

- drop rows only if All columns are NULL
- i.e. row drops only when everything is NULL

3) `df.dropna(subset=["col1"])`

- drop rows based on specific columns
- drop rows only if **col1** is NULL
- ignores NULLs in other columns

```
df.dropna(subset=["col1", "col2"])
```

4) `df.dropna(thresh=2)`

- minimum non-NULL values required (thresh)
- keeps rows that have atleast 2 non-null values

⇒ we can use them all together to...

```
df.dropna(how="any", subset=['col1'], thresh=2)
```

isNull() & isNotNull()

- They are column-level functions used to check whether a value is NULL or NOT NULL

isNULL	detect Null
isNotNull	detect not-null

1) `df.filter( col("Col1").isNULL())`
   returns rows with missing Col1 values

2) `df.filter( col("Col1").isNotNull())`
   returns only valid rows

3)
```
df.filter(
        col("Col1").isNotNull() &
        col("Col2").isNull()
)
```

4)
```
df.withColumns(
        "Col1",
        when( col("Col2").isNull(), "Missing")
        .otherwise("present")
)
```

coalesce()

- returns the first NON-NULL value from a list of columns (or) literals

- It is used for fallback/priority based NULL handling

```
df.withColumn(
        "final_col",
        coalesce( col("col1"), col("col2"))
)
```

if; col1 is NULL → uses col2 value

```
df.withColumn(
        "final",
        coalesce( col("col1"), lit(5))
)
```

```
df.withColumn(
        "final",
        coalesce(
                col("col1"),
                col("col2"),
                :
                lit(5)
        )
)
```

coalesce works per row, not per column

coalesce Example

id	Col1	Col2	Col3
1	NULL	10	100
2	5	20	30
3	NULL	NULL	300
4	NULL	NULL	NULL

```
df.withColumn(
    "result",
    coalesce(
        col("col1"),
        col("col2"),
        col("col3"),
        lit(0)
    )
)
```

How coalesce() works? (Row By Row)

Row1:

coalesce(NULL, 10, 100, 0)
↳ first_non_null = 10

Row2:

coalesce(5, 20, 30, 0)
↳ first_non_null = 5

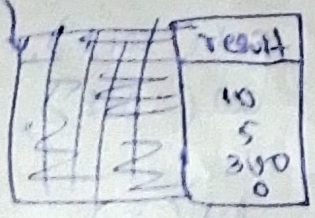
Row3:

coalesce(NULL, NULL, 300, 0)
↳ first_non_null = 300

Row4:

coalesce(NULL, NULL, NULL, 0)
↳ first_non_null = 0

∴ result



coalesce() → only NULL handling
when(otherwise()) → also to business logic

Note:

If default value is not given through lit() then at row-4 "NULL" will be printed

i.e; ⇒ coalesce(NULL, NULL, NULL) → NULL