

Expectations

# Learning Objectives

- ▶ Definition of Expectations
- ▶ SQL and Python Syntax

# Expectations

- ▶ Enforce data quality through constraints

```
CREATE OR REFRESH OBJECT <object_name>  
(CONSTRAINT <constraint_name> EXPECT(condition))
```

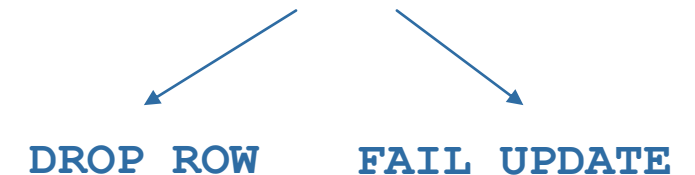
- ▶ Collect metrics on constraint violations



# Expectations

- ▶ By default, records violating constraints will be kept
- ▶ Define different actions

`(CONSTRAINT <name> EXPECT (condition) [ON VIOLATION ACTION])`



# Python Syntax

► @dp.table

**@dp.expect**(<constraint-name>, <condition>)

**@dp.expect\_or\_drop**(<constraint-name>, <condition>)

**@dp.expect\_or\_fail**(<constraint-name>, <condition>)

```
def my_table():
```

```
    # create table
```

# Example

- ▶ `CREATE OR REFRESH STREAMING TABLE my_table (  
    CONSTRAINT recent_status EXPECT (status = 'ACTIVE' AND date >= '2025-01-01'),  
    CONSTRAINT positive_value EXPECT (value > 0) ON VIOLATION DROP ROW,  
    CONSTRAINT valid_id EXPECT (id IS NOT NULL) ON VIOLATION FAIL UPDATE  
) AS SELECT * FROM STREAM(source)`
- ▶ `@dp.table  
@dp.expect("recent_status", "status = 'ACTIVE' AND date >= '2025-01-01")  
@dp.expect_or_drop("positive_value", "value > 0")  
@dp.expect_or_fail("valid_id", "id IS NOT NULL")  
def my_table():  
    spark.readStream.table("sorce")`

# Multiple Expectations

- ▶ `constraints = {"constraint-1": "cond-1", "constraint-2": "cond-2", ...}`
- ▶ `dp.expect_all(constraints)`
- ▶ `dp.expect_all_or_drop(constraints)`
- ▶ `dp.expect_all_or_fail(constraints)`

# Expectations

Action	SQL syntax	Python syntax
<b>Warn</b> (default)	EXPECT (...)	dp.expect dp.expect_all
<b>Drop</b>	EXPECT (...) ON VIOLATION DROP ROW	dp.expect_or_drop dp.expect_all_or_drop
<b>Fail</b>	EXPECT (...) ON VIOLATION FAIL UPDATE	dp.expect_or_fail dp.expect_all_or_fail