PySpark Tip ⚡

UNIT TEST PYSPARK WITH PYTEST

Mitchell van Rijkom
@MitchellvRijkom

SWIPE

PySpark Tip ⚡

CREATE SPARKSESSION FIXTURE

SparkSession is a unified entry point of a spark application

We do not want to instantiate it every time because it will slow down our tests.

Create a fixture and pass it to every test as a parameter.

tests/conftest.py

```python
import pytest
from pyspark.sql import SparkSession


@pytest.fixture(scope="session")
def spark():
    spark = (
        SparkSession
        .builder
        .master("local[*]")
        .getOrCreate()
    )
    yield spark
    spark.stop()
```

Mitchell van Rijkom
@MitchellvRijkom

SWIPE

PySpark Tip ⚡

CREATE TRANSFORMATIONS

Create a file with data transformations which will be tested!

src/etl.py

```python
from pyspark.sql.window import Window
from pyspark.sql import functions as F, Row, DataFrame

def filter_most_recent_id(df: DataFrame) -> DataFrame:
    win_spec = (
        Window.partitionBy('id')
        .orderBy(F.col('sequence_nr').desc())
    )

    return (
        df
        .withColumn("row_number", F.row_number().over(win_spec))
        .filter(F.col("row_number") == 1)
        .drop("row_number")
    )
```

Mitchell van Rijkom
@MitchellvRijkom

SWIPE

PySpark Tip ⚡

TO LIST HELPER METHOD

There is no built-in DataFrame assertion method.

If we convert our DataFrame to a list, we can check for equality and assert with another list!

tests/helpers.py

```python
# Based on:
# https://github.com/ivangeorgiev/unittest-pyspark
from typing import Dict
from pyspark.sql import Row, DataFrame

def row_as_dict(row: Row) -> Dict:
    d = row.asDict()
    for key in d:
        if isinstance(d[key], Row):
            d[key] = row_as_dict(d[key])
    return d

def as_list(df: DataFrame) -> DataFrame:
    result = df
    if isinstance(df, DataFrame):
        result = [row_as_dict(r) for r in df.collect()]
    return result
```

Mitchell van Rijkom
@MitchellvRijkom

SWIPE

PySpark Tip ⚡

CREATE & RUN TEST

Test by creating dummy input data, run transformation, and asserting the output.

If you put your code in small modular components, you can put all tests cases in a single DataFrame. It is not necessary to create a separate test for each data combination!

tests/test_etl.py

```python
from pyspark.sql import Row
from src.etl import filter_most_recent_id
from .helpers import as_list

def test_filter_most_recent_id(spark):
    # Arrange
    test_row = [
        Row(id="123", sequence_nr=1),
        Row(id="123", sequence_nr=2),
        Row(id="456", sequence_nr=None),
        Row(id="456", sequence_nr=1),
        Row(id=None, sequence_nr=1),
        Row(id=None, sequence_nr=2),
    ]
    item_df = spark.createDataFrame(test_row)

    # Act
    actual_df = item_df.transform(filter_most_recent_id)
    actual = as_list(actual_df)

    # Assert
    expect = [
        {"id": None, "sequence_nr": 2},
        {"id": "123", "sequence_nr": 2},
        {"id": "456", "sequence_nr": 1},
    ]
    assert actual == expect
```

SparkSession fixture

Arrange test data to certain condition

System under test

Assert output

```
python -m pytest -s
.

============================== 1 passed in 12.70s ==============================
```

Run tests!

Mitchell van Rijkom
@MitchellvRijkom

SWIPE

DID YOU FIND THIS TIP USEFUL?

Please, give it a like, and
share it with other people!

You can also follow me on:

social.mitchellvanrijkom.com/twitter

social.mitchellvanrijkom.com/linkedin

social.mitchellvanrijkom.com/instagram

Mitchell van Rijkom
@MitchellvRijkom