pytest.ini
==========

```
[pytest]
markers =
    slow: mark a test as slow
    transformation: mark a test as a transformation
    latest: to test the latest
```

test_retail_proj.py
==========

```
import pytest
from lib.Utils import get_spark_session
from lib.DataReader import read_customers, read_orders
from lib.DataManipulation import filter_closed_orders,
count_orders_state,filter_orders_generic
from lib.ConfigReader import get_app_config

@pytest.mark.skip("work in process")
def test_read_customers_df(spark):
    customers_count = read_customers(spark, "LOCAL").count()
    assert customers_count == 12435

@pytest.mark.skip("work in process")
def test_read_orders_df(spark):
    orders_count = read_orders(spark, "LOCAL").count()
    assert orders_count == 68883

@pytest.mark.parametrize(
"entry1,count",
[("CLOSED", 7556),
 ("PENDING_PAYMENT", 15030),
 ("COMPLETE", 22899)])

@pytest.mark.latest()
def test_check_count_df(spark,entry1,count):
    orders_df = read_orders(spark, "LOCAL")
    filtered_count = filter_orders_generic(orders_df,entry1).count()
    assert filtered_count == count

@pytest.mark.transformation()
def test_filter_closed_orders(spark):
    orders_df = read_orders(spark, "LOCAL")
    filtered_count = filter_closed_orders(orders_df).count()
    assert filtered_count == 7556
```

```
@pytest.mark.slow()
def test_read_app_config():
    config = get_app_config("LOCAL")
    assert config["orders.file.path"] == "data/orders.csv"

@pytest.mark.skip("work in process")
def test_count_orders_state(spark,expected_results):
    customers_df = read_customers(spark, "LOCAL")
    actual_results = count_orders_state(customers_df)
    assert actual_results.collect() == expected_results.collect()
```

conftest.py
==========

```
import pytest
from lib.Utils import get_spark_session

@pytest.fixture
def spark():
    "creates a spark session"
    return get_spark_session("LOCAL")

@pytest.fixture
def spark():
    "creates a spark session"
    spark_session = get_spark_session("LOCAL")
    yield spark_session
    spark_session.stop()

@pytest.fixture
def expected_results(spark):
    "gives the expected results"
    results_schema = "state string,count int"
    return spark.read \
        .format("csv") \
        .schema(results_schema) \
        .load("data/test_result/state_aggregate.csv")
```

DataManipulation.py
==========

```
from pyspark.sql.functions import *

def filter_closed_orders(orders_df):
    return orders_df.filter("order_status = 'CLOSED'")

def join_orders_customers(orders_df, customers_df):
    return orders_df.join(customers_df, "customer_id")

def count_orders_state(joined_df):
    return joined_df.groupBy('state').count()

def filter_orders_generic(orders_df,status):
    return orders_df.filter("order_status = '{}'".format(status))
```