**Simple UnitTestCase:**

1. Build a python application which can be submitted using spark-submit
2. Implement a simple unittest using the python module unittest.

* **Setup Requirements for the project:**

Ubuntu latest

Pyspark

Python

Jdk

Conda

Git

* Here, taking the main calculation of average cooking time based on the difficulty
* **Save the below function as test.py in a file**

from pyspark.sql.types import \*

from pyspark.sql.functions import \*

def transform\_data(input\_df):

transformed\_df = input\_df.groupBy('Difficulty',).agg({"TimeinMin":"avg"})

return transformed\_df

(input\_df is the the dataframe created in the main python file. This is just the example:

Eg input data and the output is as follows)

A computer screen capture

Description automatically generated with medium confidence

Text

Description automatically generated

* **Implement UnitTestCases:**

1. **Import the required modules into a file as modules.py**

import unittest

from etl.etl import transform\_data

from pyspark.sql.functions import \*

from pyspark.sql.types import \*

from datetime import datetime

from pyspark.sql import SparkSession

1. Define a class file for the unittestcase

import unittest

class SparkETLTestCase(unittest.TestCase):

@classmethod

def setUpClass(cls):

cls.spark = (SparkSession

.builder

.master("local[\*]")

.appName("Unit-tests")

.getOrCreate())

@classmethod

def tearDownClass(cls):

cls.spark.stop()

1. Prepare an input data frame similar to our source data
2. Expected dataframe which is the expected output
3. Apply the transformation on the created input dataframe
4. Assert the output to expected dataframe

import unittest

from test import \*

from pyspark.sql.types import \*

from pyspark.sql.functions import \*

def test\_etl(self):

#1. Prepare an input data frame that mimics our source data.

input\_schema = StructType([

StructField('cookTime', IntegerType()),

StructField('datePublished', StringType()),

StructField('description', StringType()),

StructField('image', StringType()),

StructField('ingredients', StringType()),

StructField('name', StringType()),

StructField('prepTime', IntegerType()),

StructField('recipeYield', StringType()),

StructField('url', StringType()),

StructField('TotalCookTime', IntegerType()),

StructField('TimeinMin', StringType()),

StructField('Difficulty', StringType())])

input\_data = [(1800, "2010-11-23", "Important note:", "http://static.the.","2 Tablespoons But..","French Onion Soup",1200,"8","http://thepioneer",3000,"50","Medium" ),

(300, "2010-11-24", "Important note:", "http://static.the.","2 Tablespoons But..","French Onion Soup",900,"8","http://thepioneer",1200,"20","Easy" ),

(900, "2010-11-24", "Important note:", "http://static.the.","2 Tablespoons But..","French Onion Soup",7200,"8","http://thepioneer",8100,"135","Hard" )]

input\_df = self.spark.createDataFrame(data=input\_data, schema=input\_schema)

#2. Prepare an expected data frame which is the output that we expect.

expected\_schema = StructType([

StructField('Difficulty', StringType(), True),

StructField('avg(TimeinMin)', Double(), True)

])

expected\_data = [("Easy", 19.625),

("Medium", 40.0),

("Hard", 174.481)]

expected\_df = self.spark.createDataFrame(data=expected\_data, schema=expected\_schema)

df5 = transform\_data(input\_df)

#4. Assert the output of the transformation to the expected data frame.

field\_list = lambda fields: (fields.name, fields.dataType, fields.nullable)

fields1 = [\*map(field\_list, df.schema.fields)]

fields2 = [\*map(field\_list, expected\_df.schema.fields)]

# Compare schema of df and expected\_df

res = set(fields1) == set(fields2)

# assert

self.assertTrue(res)

# Compare data in df and expected\_df

self.assertEqual(sorted(expected\_df.collect()), sorted(df.collect()))

Run the unittest case as below:

python -m unittest -v