















Algunas pistas para la Tarea 2

Contenido

A. Qué archivos podría tener en la carpeta:	1
B. Qué podría contener el archivo conftest:	2
C. Cómo se podría organizar el archivo main:	2
D. Cómo podría iniciar el archivo main:	2
E. Cómo podría iniciar el archivo functions:	3
F. Cómo podría iniciar el archivo unit_test:	3
G. Cómo podría verse un json:	3

A. Qué archivos podría tener en la carpeta:

-  metricas.csv
-  total_ingresos.csv
-  total_viajes.csv
-  build_image
-  conftest
-  Dockerfile
-  main
-  repartidor1
-  repartidor2
-  run_image
-  solucion_tarea2
-  tarea2_functions
-  tarea2_main
-  test_tarea2

B. Qué podría contener el archivo conftest:

```
conftest.py X
C: > LuisAlex > CURSO_BigData > Lec2 > Tarea 1 > solucion_tarea1 > conftest.py > ...
1  import pytest
2
3  from pyspark.sql import SparkSession
4
5
6  @pytest.fixture(scope="module")
7  def spark_session():
8      """A fixture to create a Spark Context to reuse across tests."""
9      s = SparkSession.builder.appName('pytest-local-spark').master('local') \
10         .getOrCreate()
11
12     yield s
13
14     s.stop()
15
```

C. Cómo se podría organizar el archivo main:

- Imports
- Preparación del ambiente
- Cargar los datos
- Generar dataframes y calcular
- Generar los archivos .csv

D. Cómo podría iniciar el archivo main:

```
1  from pyspark.sql import SparkSession
2  from tarea2_functions import read_files_json, viajes, ingresos, top_driver_km, top_driver_ingresos, percentil
3
4  #Creamos un dataframe a partir de los archivos json que encuentre en el mismo folder de ejecucion
5  dataframe = read_files_json()
6  dataframe.show()
~
```

E. Cómo podría iniciar el archivo functions:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, FloatType, StructField, DateType, StructType, StringType, TimestampType
import pyspark.sql.functions as f
from pyspark.sql import DataFrameStatFunctions as statFunc
from pyspark.sql.functions import rank, col, round, explode, lit
from pyspark.sql import Window
import json

spark = SparkSession.builder.appName("Didi drivers").getOrCreate()

...

Funcion para leer los json files correspondientes a la informacion de los conductores con las rutas recorridas.
Se crearon 5 archivos json a modo de prueba pero la funcion lee todos los que encuentre en el folder con extension .json
y los convierte a un dataframe en conjunto
...

def read_files_json():
    multiline_df = spark.read.option("multiline", "true").json("*.json").select("identificador", explode("viajes")).\
        withColumn("identificador", col("identificador")).\
```

F. Cómo podría iniciar el archivo unit_test:

```
from tarea2_functions import dataframe_ingresos, viajes, ingresos, top_driver_ingresos, top_driver_km, percentil, df_metricas
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, FloatType, StructField, DateType, StructType, StringType, TimestampType

...

Prueba para evaluar la funcion dataframe_ingresos quien crea una columna adicional llamada Ingresos y que es el resultado de multiplicar
los kilometros recorridos por el precio por kilometro
...

def test_df_ingresos(spark_session):
    df = [('20122021', '10101', '10102', 10.21, 800.0),
          ('20122021', '10103', '10104', 15.3, 750.0),
          ('20122021', '10105', '10106', 8.4, 900.0)]
```

G. Cómo podría verse un json:

```
1  {
2      "identificador": "1",
3      "viajes": [
4          {
5              "codigo_postal_origen": "1000",
6              "codigo_postal_destino": "1002",
7              "kilometros": 2.8,
8              "precio_kilometro": 550
9          },
10         {
11             "codigo_postal_origen": "3000",
12             "codigo_postal_destino": "3001",
13             "kilometros": 150.3,
14             "precio_kilometro": 300
15         }
16     ]
17 }
```