Manual Tarea 2

Estudiante: María Yorleni Alfaro Alfaro

En el presente documento se encuentran las instrucciones respectivas para ejecutar el programa principal.

Pasos para la ejecución:

1. Primero se debe descomprimir el archivo Tarea2.zip entregado en la tarea.

   Los siguientes son los archivos que se encuentran en dicha carpeta y son necesarios para la correcta ejecución:

   - **tarea2.py**: es el programa principal, desde el cual se realiza el llamado a las diferentes funciones.
   - **tarea2_funciones.py**: en este archivo se encuentra el código de cada una de las funciones desarrolladas (que son llamadas desde el programa principal "tarea2.py")
   - **test_tarea2.py**: contiene una serie de pruebas unitarias que permitan corroborar la correctitud de las diferentes funciones internas al programa.
   - **Archivos JSON**: contienen todos los viajes realizados por cada conductor(a) de Diber.
     - viajes_Diber1.json
     - viajes_Diber2.json
     - viajes_Diber3.json
     - viajes_Diber5.json
     - viajes_Diber5.json
   - **Archivos de configuración**:
     - __init__.py
     - conftest.py
     - Dockerfile

2. Abrir Power Shell o línea de comandos
3. Navegar al directorio donde se descomprimieron los archivos del código fuente
4. Realizar el build de la imagen con el comando: **docker build --tag tarea2 .**
5. Ejecutar la imagen del Docker con el comando: **docker run -i -t tarea2 /bin/bash**

   Nota importante: Si desea revisar en su máquina local las carpetas y archivos csv que se van a generar con la ejecución del programa principal, se debe ejecutar el siguiente comando en lugar del anterior:
   docker run -i -v "C:\tarea2:/src" -t tarea2 /bin/bash
   donde "**C:\tarea2**" corresponde a la ruta y directorio donde se descomprimió la tarea en el punto 1. Esto mapea el contenedor con dicho directorio, por lo que las carpetas y archivos que se creen en el contenedor se mostrarán en ese directorio en su máquina local.

6. Para la ejecución del programa principal, ejecutar el siguiente comando:
   "**spark-submit tarea2.py viajes_Diber*.json**".

   Como producto de la ejecución anterior, se crearán (o se sobreescribirán) en el **contenedor**, en la misma carpeta donde se encuentra el programa principal, las siguientes 3 carpetas (que contienen los archivos .csv solicitados en el enunciado de la tarea):
   - **total_viajes**: contiene un archivo .csv con 3 columnas que representan el código postal, si es origen o destino y la cantidad total de viajes para ese código postal como destino u origen (ignorar los archivos adicionales que se generan en dicha carpeta, cuya extensión no es .csv).
   - **total_ingresos**: contiene un archivo .csv con 3 columnas que representan el código postal, si es origen o destino y la cantidad de dinero generado en ingresos para ese código postal como destino u origen (ignorar los archivos adicionales que se generan en dicha carpeta, cuya extensión no es .csv).
   - **metricas**: contiene un archivo .csv con 2 columnas que representan el tipo de métrica y su valor (ignorar los archivos adicionales que se generan en dicha carpeta, cuya extensión no es .csv).

   Nota importante: en la carpeta descomprimida que contiene la tarea, se encuentran dichas carpetas (total_viajes, total_ingresos y metricas) las cuales contienen los archivos csv generados en la ejecución que se realizó cuando se estaba desarrollando el programa. Dado que las carpetas y los archivos csv generados con la ejecución del programa principal quedan en el contenedor, si desea revisarlos en la máquina local, debió haber ejecutado la segunda instrucción de **docker run** que se indicó en el punto 5 (con lo cual dichas carpetas serán sobreescritas en su máquina local con los nuevos resultados).

7. Para la ejecución de las pruebas, ejecutar el siguiente comando:
   "**pytest test_tarea2.py -v**".

   Con la ejecución del comando anterior se muestra el resultado de cada una de las pruebas.

Pruebas realizadas:

- **Sección "Total de viajes":**
  - **test_total_viajes_por_codigo_postal_origen_1_viaje_por_codigo_postal**: prueba la función de obtener el total de viajes por código postal origen, cuando se tiene un viaje registrado por cada código postal; se obtiene el registro de cada código postal origen con una cantidad total de viajes de 1.
  - **test_total_viajes_por_codigo_postal_origen_varios_viajes_por_codigo_postal**: prueba la función de obtener el total de viajes por código postal origen, cuando se tienen varios viajes registrados por cada código postal; se obtiene el registro de cada código postal origen con la respectiva cantidad total de viajes.
  - **test_total_viajes_por_codigo_postal_origen_mismo_viaje_varias_veces**: prueba la función de obtener el total de viajes por código postal origen, cuando se tiene un

mismo viaje registrado varias veces para un mismo código postal origen; se obtiene el registro del código postal origen con la cantidad total de viajes.

- **test_total_viajes_por_codigo_postal_origen_kilometros_negativos_cero_null**: prueba la función de obtener el total de viajes por código postal origen, cuando los kilómetros vienen en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con kilómetros en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de viajes).
- **test_total_viajes_por_codigo_postal_origen_precioKm_negativo_cero_null**: prueba la función de obtener el total de viajes por código postal origen, cuando el precio_kilometro viene en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con precio_kilometro en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de viajes).
- **test_total_viajes_por_codigo_postal_origen_invalido**: prueba la función de obtener el total de viajes por código postal origen, cuando el codigo_postal_origen viene en null; en este caso se excluyen los registros que vienen con codigo_postal_origen en null (no son tomados en cuenta a la hora de obtener el total de viajes).
- **test_total_viajes_por_codigo_postal_destino_1_viaje_por_codigo_postal**: prueba la función de obtener el total de viajes por código postal destino, cuando se tiene un viaje registrado por cada código postal; se obtiene el registro de cada código postal destino con una cantidad total de viajes de 1.
- **test_total_viajes_por_codigo_postal_destino_varios_viajes_por_codigo_postal**: prueba la función de obtener el total de viajes por código postal destino, cuando se tienen varios viajes registrados por cada código postal; se obtiene el registro de cada código postal destino con la respectiva cantidad total de viajes.
- **test_total_viajes_por_codigo_postal_destino_mismo_viaje_varias_veces**: prueba la función de obtener el total de viajes por código postal destino, cuando se tiene un mismo viaje registrado varias veces para un mismo código postal destino; se obtiene el registro del código postal destino con la cantidad total de viajes.
- **test_total_viajes_por_codigo_postal_destino_kilometros_negativos_cero_null**: prueba la función de obtener el total de viajes por código postal destino, cuando los kilómetros vienen en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con kilómetros en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de viajes).
- **test_total_viajes_por_codigo_postal_destino_precioKm_negativo_cero_null**: prueba la función de obtener el total de viajes por código postal destino, cuando el precio_kilometro viene en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con precio_kilometro en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de viajes).
- **test_total_viajes_por_codigo_postal_destino_invalido**: prueba la función de obtener el total de viajes por código postal destino, cuando el codigo_postal_destino viene en null; en este caso se excluyen los registros que vienen con codigo_postal_destino en null (no son tomados en cuenta a la hora de obtener el total de viajes).

- **test_unir_dataframes_total_viajes_por_codigo_postal_origen_destino_verifica_datos**: prueba la función unir_dataframes cuando se une los dataframes del total de viajes por código postal origen y total de viajes por código postal destino; se obtiene un solo dataframe con la unión de los registros de ambos dataframes.
- **test_unir_dataframes_total_viajes_por_codigo_postal_origen_destino_verifica_cantidad_registros**: prueba la función unir_dataframes cuando se une los dataframes del total de viajes por código postal origen y total de viajes por código postal destino, verificando la cantidad de registros de cada uno de los dataframes; verifica que la suma de la cantidad de registros de ambos dataframes sea igual a la cantidad de registros del dataframe generado con la unión.

- **Sección "Total de ingresos":**
  - **test_total_ingresos_por_codigo_postal_origen_1_viaje_por_codigo_postal**: prueba la función de obtener el total de ingresos por código postal origen, cuando se tiene solo un viaje registrado por cada código postal; se obtiene el registro de cada código postal origen con el total de ingresos (kilometros * precio_kilometro) por ese viaje.
  - **test_total_ingresos_por_codigo_postal_origen_varios_viajes_por_codigo_postal** : prueba la función de obtener el total de ingresos por código postal origen, cuando se tienen varios viajes registrados por cada código postal; se obtiene el registro de cada código postal origen con su total de ingresos (suma de todos los ingresos por viaje -el ingreso de cada viaje corresponde a kilometros * precio_kilometro-).
  - **test_total_ingresos_por_codigo_postal_origen_mismo_viaje_varias_veces**: prueba la función de obtener el total de ingresos por código postal origen, cuando se tiene un mismo viaje registrado varias veces para un mismo código postal origen; se obtiene el registro del código postal origen con su total de ingresos (suma de todos los ingresos por cada uno de esos viajes).
  - **test_total_ingresos_por_codigo_postal_origen_kilometros_negativos_cero_null**: prueba la función de obtener el total de ingresos por código postal origen, cuando los kilómetros vienen en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con kilómetros en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de ingresos).
  - **test_total_ingresos_por_codigo_postal_origen_precioKm_negativo_cero_null**: prueba la función de obtener el total de ingresos por código postal origen, cuando el precio_kilometro viene en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con precio_kilometro en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de ingresos).
  - **test_total_ingresos_por_codigo_postal_origen_invalido**: prueba la función de obtener el total de ingresos por código postal origen, cuando el codigo_postal_origen viene en null; en este caso se excluyen los registros que vienen con codigo_postal_origen en null (no son tomados en cuenta a la hora de obtener el total de ingresos).
  - **test_total_ingresos_por_codigo_postal_destino_1_viaje_por_codigo_postal**: prueba la función de obtener el total de ingresos por código postal destino, cuando

se tiene solo un viaje registrado por cada código postal; se obtiene el registro de cada código postal destino con el total de ingresos (kilometros * precio_kilometro) por ese viaje.

- **test_total_ingresos_por_codigo_postal_destino_varios_viajes_por_codigo_postal**: prueba la función de obtener el total de ingresos por código postal destino, cuando se tienen varios viajes registrados por cada código postal; se obtiene el registro de cada código postal destino con su total de ingresos (suma de todos los ingresos por viaje -el ingreso de cada viaje corresponde a kilometros * precio_kilometro-).
- **test_total_ingresos_por_codigo_postal_destino_mismo_viaje_varias_veces**: prueba la función de obtener el total de ingresos por código postal destino, cuando se tiene un mismo viaje registrado varias veces para un mismo código postal destino; se obtiene el registro del código postal destino con su total de ingresos (suma de todos los ingresos por cada uno de esos viajes).
- **test_total_ingresos_por_codigo_postal_destino_kilometros_negativos_cero_null**: prueba la función de obtener el total de ingresos por código postal destino, cuando los kilómetros vienen en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con kilómetros en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de ingresos).
- **test_total_ingresos_por_codigo_postal_destino_precioKm_negativo_cero_null**: prueba la función de obtener el total de ingresos por código postal destino, cuando el precio_kilometro viene en 0, null o con valores negativos; en este caso se excluyen los registros que vienen con precio_kilometro en 0, null o valores negativos (no son tomados en cuenta a la hora de obtener el total de ingresos).
- **test_total_ingresos_por_codigo_postal_destino_invalido**: prueba la función de obtener el total de ingresos por código postal destino, cuando el codigo_postal_destino viene en null; en este caso se excluyen los registros que vienen con codigo_postal_destino en null (no son tomados en cuenta a la hora de obtener el total de ingresos).
- **test_unir_dataframes_total_ingresos_por_codigo_postal_origen_destino_verifica_datos**: prueba la función unir_dataframes cuando se une los dataframes del total de ingresos por código postal origen y total de ingresos por código postal destino; se obtiene un solo dataframe con la unión de los registros de ambos dataframes.
- **test_unir_dataframes_total_ingresos_por_codigo_postal_origen_destino_verifica_cantidad_registros**: prueba la función unir_dataframes cuando se une los dataframes del total de ingresos por código postal origen y total de ingresos por código postal destino, verificando la cantidad de registros de cada uno de los dataframes; verifica que la suma de la cantidad de registros de ambos dataframes sea igual a la cantidad de registros del dataframe generado con la unión.

- **Sección "Métricas":**
  - **test_metrica_persona_con_mas_kilometros_1_viaje_por_persona:** prueba la función de obtener métrica de persona con más kilómetros cuando se tiene solo un

viaje registrado por cada persona; se obtiene el nombre de la métrica y el identificador de la persona con más kilómetros registrados.

- **test_metrica_persona_con_mas_kilometros_varios_viajes_por_persona:** prueba la función de obtener métrica de persona con más kilómetros cuando se tienen varios viajes registrados por cada persona; se suman los kilómetros registrados por cada viaje para cada persona y se obtiene el nombre de la métrica y el identificador de la persona con más kilómetros registrados.
- **test_metrica_persona_con_mas_kilometros_personas_empatadas:** prueba la función de obtener métrica de persona con más kilómetros cuando se tienen varias personas con los mismos viajes registrados, es decir, con la misma cantidad total de kilómetros; en este caso se elige la persona con identificador menor, por lo tanto, se obtiene el nombre de la métrica y el identificador de la persona con más kilómetros registrados (con menor identificador en caso de empate).
- **test_metrica_persona_con_mas_ingresos_1_viaje_por_persona:** prueba la función de obtener métrica de persona con más ingresos cuando se tiene solo un viaje registrado por cada persona; se obtiene el nombre de la métrica y el identificador de la persona con más ingresos (kilometros * precio_kilometro) registrados.
- **test_metrica_persona_con_mas_ingresos_varios_viajes_por_persona:** prueba la función de obtener métrica de persona con más ingresos cuando se tienen varios viajes registrados por cada persona; se obtiene el nombre de la métrica y el identificador de la persona con más ingresos (suma los ingresos por cada uno de esos viajes y obtiene la persona con mayor cantidad de ingresos).
- **test_metrica_persona_con_mas_ingresos_personas_empatadas:** prueba la función de obtener métrica de persona con más ingresos cuando se tienen varias personas con los mismos viajes registrados (mismos datos de cada viaje), es decir, con los mismos ingresos totales; en este caso se elige la persona con identificador menor, por lo tanto, se obtiene el nombre de la métrica y el identificador de la persona con más ingresos registrados (con menor identificador en caso de empate).
- **test_metrica_percentil_25:** prueba la función de obtener el percentil 25 en un dataset de 4 personas diferentes, por lo tanto el percentil 25 corresponde a la fila 1 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 1 que representa el percentil 25.
- **test_metrica_percentil_25_con_redondeo:** prueba la función de obtener el percentil 25 en un dataset de 10 personas diferentes, por lo tanto el percentil 25 corresponde a 2.5 por lo que se redondea a la fila 3 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 3 que representa el percentil 25.
- **test_metrica_percentil_50:** prueba la función de obtener el percentil 50 en un dataset de 10 personas diferentes, por lo tanto el percentil 50 corresponde a la fila 5 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 5 que representa el percentil 50.

- **test_metrica_percentil_50_con_redondeo:** prueba la función de obtener el percentil 50 en un dataset de 11 personas diferentes (el dataset contiene más registros porque vienen personas con más de un viaje), por lo tanto el percentil 50 corresponde a 5.5 por lo que se redondea a la fila 6 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 6 que representa el percentil 50.
- **test_metrica_percentil_75:** prueba la función de obtener el percentil 75 en un dataset de 12 personas diferentes (el dataset contiene más registros porque vienen personas con más de un viaje), por lo tanto el percentil 75 corresponde a la fila 9 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 9 que representa el percentil 75.
- **test_metrica_percentil_75_con_redondeo:** prueba la función de obtener el percentil 75 en un dataset de 10 personas diferentes (el dataset contiene más registros porque vienen personas con más de un viaje), por lo tanto el percentil 75 corresponde a 7.5 por lo que se redondea a la fila 8 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 8 que representa el percentil 75.
- **test_metrica_percentil_10:** prueba la función de obtener el percentil 10 en un dataset de 10 personas diferentes, por lo tanto el percentil 10 corresponde a la fila 1 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 1 que representa el percentil 10.
- **test_metrica_percentil_90_con_redondeo:** prueba la función de obtener el percentil 90 en un dataset de 11 personas diferentes (el dataset contiene más registros porque vienen personas con más de un viaje), por lo tanto el percentil 90 corresponde a 9.9 por lo que se redondea a la fila 10 (sobre los datos ya ordenados de personas de menor a mayor cantidad de ingresos); se obtiene el nombre de la métrica y el valor de ingresos de la fila 10 que representa el percentil 90.
- **test_metrica_percentil_110:** prueba la función de obtener un percentil mayor a 100 (en este caso 110); la función lo que hace es calcular el percentil 100; por lo tanto se obtiene el nombre de la métrica "percentil_100" y el valor de ingresos que corresponde al percentil 100.
- **test_metrica_percentil_negativo:** prueba la función de obtener un percentil menor a 0 (en este caso -1); la función lo que hace es calcular el percentil 0; por lo tanto se obtiene el nombre de la métrica "percentil_0" y el valor de ingresos que corresponde al percentil 0.
- **test_metrica_codigo_postal_origen_con_mas_ingresos_1_viaje_por_codigo_postal:** prueba la función de obtener métrica del código postal origen con más ingresos cuando se tiene solo un viaje registrado por cada código postal origen; se obtiene el nombre de la métrica y el código postal origen con más ingresos (kilometros * precio_kilometro) registrados.
- **test_metrica_codigo_postal_origen_con_mas_ingresos_varios_viajes_por_codigo_postal:** prueba la función de obtener métrica del código postal origen con más

ingresos cuando se tienen varios viajes registrados por cada código postal origen; se obtiene el nombre de la métrica y el código postal origen con más ingresos (suma los ingresos por cada uno de esos viajes y obtiene el código postal origen con mayor cantidad de ingresos).

- **test_metrica_codigo_postal_origen_con_mas_ingresos_codigos_postales_empatados:** prueba la función de obtener métrica del código postal origen con más ingresos cuando se tienen códigos postales origen con la misma información (mismos datos de cada viaje), es decir, con los mismos ingresos totales; en este caso se elige el código postal menor, por lo tanto, se obtiene el nombre de la métrica y el código postal origen con más ingresos registrados (con menor código en caso de empate).
- **test_metrica_codigo_postal_origen_con_mas_ingresos_registros_con_datos_invalidos:** prueba la función de obtener métrica del código postal origen con más ingresos, cuando el codigo_postal_origen viene en null; en este caso se obtiene el nombre de la métrica y el código postal origen con más ingresos registrados (excluyendo los registros que vienen con codigo_postal_origen en null, dado que estos no son tomados en cuenta a la hora de obtener el total de ingresos).
- **test_metrica_codigo_postal_destino_con_mas_ingresos_1_viaje_por_codigo_postal:** prueba la función de obtener métrica del código postal destino con más ingresos cuando se tiene solo un viaje registrado por cada código postal destino; se obtiene el nombre de la métrica y el código postal destino con más ingresos (kilometros * precio_kilometro) registrados.
- **test_metrica_codigo_postal_destino_con_mas_ingresos_varios_viajes_por_codigo_postal:** prueba la función de obtener métrica del código postal destino con más ingresos cuando se tienen varios viajes registrados por cada código postal destino; se obtiene el nombre de la métrica y el código postal destino con más ingresos (suma los ingresos por cada uno de esos viajes y obtiene el código postal destino con mayor cantidad de ingresos).
- **test_metrica_codigo_postal_destino_con_mas_ingresos_codigos_postales_empatados:** prueba la función de obtener métrica del código postal destino con más ingresos cuando se tienen códigos postales destino con la misma información (mismos datos de cada viaje), es decir, con los mismos ingresos totales; en este caso se elige el código postal menor, por lo tanto, se obtiene el nombre de la métrica y el código postal destino con más ingresos registrados (con menor código en caso de empate).
- **test_metrica_codigo_postal_destino_con_mas_ingresos_registros_con_datos_invalidos:** prueba la función de obtener métrica del código postal destino con más ingresos, cuando el codigo_postal_ destino viene en null; en este caso se obtiene el nombre de la métrica y el código postal destino con más ingresos registrados (excluyendo los registros que vienen con codigo_postal_ destino en null, dado que estos no son tomados en cuenta a la hora de obtener el total de ingresos).
- **test_unir_dataframes_metricas_verifica_datos**: prueba la función unir_dataframes_metricas cuando se unen los dataframes de cada una de las

métricas; se obtiene un solo dataframe con la unión de los registros de los 7 dataframes de métricas.

- **test_unir_dataframes_metricas_verifica_cantidad_registros**: prueba la función unir_dataframes_metricas cuando se une los dataframes de cada una de las métricas, verificando la cantidad de registros de cada uno de los dataframes; verifica que la suma de la cantidad de registros de cada dataframe sea igual a la cantidad de registros del dataframe generado con la unión.

Restricciones del programa:

- Se asume que todos los archivos contienen JSON válido
- Se asume que tanto los "kilómetros" como el "precio_kilometro" son valores numéricos (no vienen en texto, con espacios, etc).
- Se asume que el atributo "identificador" de cada archivo json es un valor numérico (y que todo valor numérico es válido).
- Decisiones tomadas:
  - En todas las funciones (al obtener el total de viajes por código postal origen y destino, obtener total de ingresos por código postal origen y destino, así como al obtener las métricas) se excluyen los registros con kilómetros o precio_kilometro negativos, en null o en 0.
  - Al obtener el total de viajes por código postal origen y destino, obtener total de ingresos por código postal origen y destino, así como al obtener las métricas de código postal origen y destino con más ingresos, se excluyen los registros con un código postal inválido (0, negativo, null).
  - A la hora de obtener las métricas de persona con más kilómetros y persona con más ingresos, si hay un empate se elige la persona con menor identificador.
  - A la hora de calcular el percentil, el ordenamiento de las personas se hace por total de ingresos (ascendente) e identificador (ascendente).
  - A la hora de calcular el percentil, si el percentil enviado como parámetro es mayor a 100, se obtiene el percentil 100 y si es menor a 0 se obtiene el percentil 0.
  - A la hora de obtener las métricas de código postal origen con más ingresos y código postal destino con más ingresos, si hay un empate se elige el que tenga el menor código.