Descargar programa

```
git clone
```

Como ejecutar el programa:
Importar a la carpeta raíz/general los datos del archivo “storage” como se ven en la imagen. Instalar los requirements.txt con el comando:

```
pip install -r requirements.txt
```

Ahora ya nos podemos situar en la carpeta “src” y ejecutar el archivo “main.py” con nuestro interprete de Python.

Para abordar el ejercicio planteado en el archivo “*202403-TechnicalTest_DEInten.pdf*” de titulo “*Prueba Técnica | Data Engineering Intern*” se ha desarrollado un programa en Python que suple los requisitos solicitados.

El código del programa se encuentra en la carpeta “*src*”.

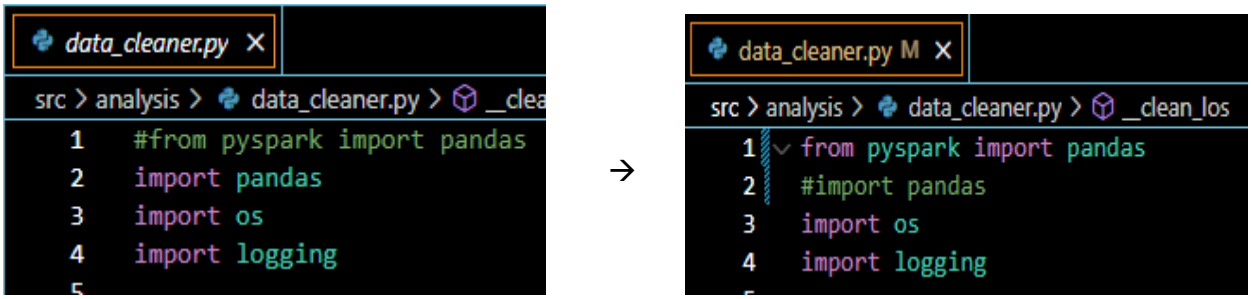
Luego dentro “*src*”, existen las subcarpetas:

- main.py (Entrypoint del programa)
- analysis.
  - data_cleaner.py (Funciones para limpiar datasets)
  - feature_engineering.py (Funciones para enriquecer datasets)
  - graph_creator.py (Funciones para crear gráficos)
- downloaders
  - simualtor
    - API_Simulator.py (Simulador de endpoint de BEONx)
  - external_data.py (Funciones para descargar datos del enpoint)
  - internal_data.py (Funciones para descargar datos del AWS S3)
- uploaders
  - upload_data.py (Funciones para subir datos a AWS S3)

El flujo del programa se esquematiza de la siguiente manera:

1. Se comienza por la descarga de ficheros. csv de un endpoint simulado con un retraso de un segundo para equipararse a un entrono real con trafico de red. Para cumplir con el requisito de no superar los 7 segundo a la hora de descargar los datos, se ha implementado un pool de hilos autogestionado en la función “download_data_from_simulator” dentro del archivo “external_data.py” de la carpeta “*downloaders*”.
   Esta función guarda los archivos descargados en la carpeta **downloads**.

2. Una vez descargados los .csv con los datos relativos a los check_in se limpian buscando valores incoherentes. Esto se realiza en la función “clean_downloaded_data” dentro del archivo “*data_cleaner.py*” de la carpeta “*analysis*”. Para esta fase se ha usado el módulo de pandas. Antes se usaba la API de PySpark que nos permite aprovechar el procesamiento distribuido, pero debido a que no dispongo de una maquina muy potente el rendimiento era inferior al de pandas. De igual manera, para activar PySpark, basta con modificar el archivo “*data_cleaner.py*” de la siguiente manera:



Esta función guarda los archivos limpiados en la carpeta **clean_data**.

3. Después de limpiarse los datos se aplica *feature_engineering*. Todo este proceso se ha implementado en la función "*feature_downloaded_data*" dentro del archivo "*feature_engineering.py*" en la carpeta "*analysis*". Las principales operaciones que se hacen son:
    a. Normalización de la columna "rate_price".
    b. Descomposición en año, mes y día de las columnas "check_in" y "check_out"
    c. Creación de la columna "season".
    d. Factorización de las columnas "meal_plan", "entity_type" y "season".

    Esta función guarda los archivos procesados en la carpeta **feature_data**

4. Una vez añadidas las nuevas columnas, subimos los dataset al servicio S3 de AWS que actuará como datalake de nuestra aplicación. Estos se realiza en la función "*upload_data_to_datalake*" dentro del archivo "*upload_data.py*" en la carpeta "*uploaders*" Con esto cumplimos el requisito de poseer un almacenamiento común y distribuido a modo de histórico, además de que es una solución optima al ofrecer sistemas de permisos en base a los recursos que estén subidos para dar libre acceso a los científicos de datos.
    Al principio se usaba Hadoop, pero debido a las fechas establecidas para completar esta tarea y a la falta de conocimiento técnico para realizar un desarrollo optima con esta plataforma no era suficiente, se busco la solución una alternativa, siendo esta S3 de AWS. Esta decisión se tomo en base a los requisitos ofertados en la oferta de trabajo de BEONx. Esto también aplica al uso de PySpark cuando se procesaban los datasets.

5. Una vez subidos, se procede a descargar de nuevo los datos de la nube S3 por medio de la función "*download_data_from_datalake*" dentro del archivo "*internal_data.py*" en la carpeta "*downloaders*".
    Esta función guarda los archivos descargados en la carpeta **s3_downloads**.

6. Una vez descargados los archivos de la nube S3 se ejecuta un análisis grafico con la función "create_graph" dentro del archivo "*graph_creator.py*" en la carpeta "*analysis*" para cumplir con otro de los requisitos solicitados.
    Esta función guarda los gráficos creados en la carpeta **graphs**

    Explicarlo todo por medio de un documento de texto limita mucho la descripción de detalles y del análisis desarrollado para llevar a cabo el proyecto. Dentro del código existen comentarios para prácticamente todas las líneas. Si tenéis alguna duda o necesitáis algún detalle, porfavor no dudéis en consultármelo.
    watzvictor1@gmail.com
    +34 674 876 056