

Prueba técnica para Ingeniería de Datos Jikkosoft

Documentación

Respuesta al reto

Iniciaré por presentar el diagrama para el pipeline propuesto, donde utilicé Python y MySQL.



Requerimientos:

1. Descarga la carpeta comprimida que contiene los datos y déjalo en una carpeta exclusiva para este reto.

La carpeta es descargada y en ella misma montamos todo el proyecto, organizando los .py en orden de ejecución. Conservando los .txt en una carpeta propia, y utilizando los .csv en el mismo proyecto.

Nombre	Estado	Fecha de modificación	Tipo	Tamaño
📁 __pycache__	✓ ⓘ	25/01/2025 13:37	Carpeta de archivos	
📁 02_dataset	✓ ⓘ	24/01/2025 17:25	Carpeta de archivos	
📄 database.py	✓ ⓘ	25/01/2025 13:18	Python.File	1 KB
📄 estadisticas	✓ ⓘ	26/01/2025 9:15	Documento de texto	3 KB
📄 main.py	✓ ⓘ	26/01/2025 9:15	Python.File	4 KB
📄 maximos	✓ ⓘ	24/01/2025 9:02	Archivo de valores separ...	1 KB
📄 minimos	✓ ⓘ	23/01/2025 16:24	Archivo de valores separ...	1 KB
📄 process_csv.py	✓ ⓘ	25/01/2025 13:18	Python.File	1 KB
📄 process_txt.py	✓ ⓘ	25/01/2025 13:18	Python.File	2 KB
📄 tarifa_por_destino	✓ ⓘ	23/01/2025 16:24	Archivo de valores separ...	1 KB
📄 utils.py	✓ ⓘ	25/01/2025 13:19	Python.File	2 KB

2. Construye un Pipeline que sea capaz de:

- a) Cargar todos los archivos .TXT (El pipeline no debe contener todo el conjunto de datos, es decir, los cinco archivos al mismo tiempo en memoria en cualquier momento).

Con la función `read_txt_in_batches` se asegura la carga de los archivos .txt.

- b) Cada Micro Batch debe contener como máximo 1000 registros.

Establecido el máximo en 1000 registros, el `batch_size` se establece en lotes de 500 registros, lo anterior dado que al analizar dataset-1 este contiene 500 registros, y los dataset en conjunto conforman 18500 registros, por lo que se diseña el pipeline para cargarlos en lotes de 500 registros.

La función `fill_missing_values` asegura que las `tarifas_por_destino.csv` que no contienen un estrato como, comercial, industrial, especial y otros, tarifados al 1.7% les asignen el estrato 7, y para el caso de oficial el estrato 4 ya que el residencial estrato 4, tarifa al 1%. En resumen, se evitan los null, o datos vacíos, en este diseño, y asegura que las columnas tengan un tipo de dato definido.

```
02_process_txt.py > ...
1  # Funciones para manejar los archivos .txt.
2
3  import pandas as pd
4  import os
5
6  def read_txt_in_batches(folder_path, batch_size=500):
7      for file_name in os.listdir(folder_path):
8          if file_name.endswith('.txt'):
9              file_path = os.path.join(folder_path, file_name)
10             for chunk in pd.read_csv(file_path, sep=',', chunksize=batch_size):
11                 chunk.rename(columns={"año": "year", "Destino": "destino"}, inplace=True)
12                 yield chunk
13
14 def fill_missing_values(batch):
15     # Rellenar valores faltantes de estrato según destino
16     batch.loc[(batch["destino"].isin(["Comercial", "Industrial", "Especial", "Otros"])) & (batch["estrato"].isnull()), "estrato"] = 7
17     batch.loc[(batch["destino"] == "Oficial") & (batch["estrato"].isnull()), "estrato"] = 4
18
19     # Rellenar valores faltantes de tarifa según destino y estrato
20     batch.loc[(batch["destino"].isin(["Comercial", "Industrial", "Especial", "Otros"])) & (batch["tarifa"].isnull()), "tarifa"] = 0.017
21     batch.loc[(batch["destino"] == "Oficial") & (batch["tarifa"].isnull()), "tarifa"] = 0.01
22
23     return batch
24
```

- c) Almacena los datos de los archivos .TXT en una base de datos de tu elección (ejemplo: PostgreSQL, MySQL, MongoDB, etc). El diseño de esta base de datos dependerá de ti, crea la tabla o tablas que creas necesarias con el esquema que creas es adecuado, pero ten presente que todos los .TXT deben ir en la misma base de datos.

El pipeline diseñado almacena los datos en una base denominada Jikkosoft, en la tabla consumos, puede apreciar el entorno workbench de MySQL, con los datos cargados, 18500 registros.

Navigator SQL File 3*

SCHEMAS

Filter objects

- jikkosoft
 - Tables consumos
 - Views
 - Stored Procedures
 - Functions
- sakila
- sys
- ventas
- world

1
2 • SELECT * FROM jikkosoft.consumos;
3 • SELECT COUNT(*) FROM consumos;
4

Result Grid | Filter Rows: COUNT(*)
18500

1
2 • SELECT * FROM jikkosoft.consumos;
3

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch row

id	year	destino	estrato	consumo	tasa_calculada	minimo	maximo
000473c0-afe3-4f9d-b2b7-74b3ab055d40	2024	Industrial	7	17699957.05	300899.26985	2353.25	2002145.10
0006aee3-1417-45c4-a510-6a4b45a5c438	2023	Oficial	4	15806269.36	158062.69360	2120.60	0.00
00070517-1b65-4746-ae22-3620945768bc	2023	Oficial	4	13548409.43	135484.09430	2120.60	0.00
0007bb47-d3d6-409f-af77-0c16e759892c	2024	Comercial	7	2141786.16	36410.36472	2353.25	1647275.00
000a5b57-b93c-428c-a161-9f1a7ac7e341	2023	Otros	7	14910076.35	253471.29795	2120.60	0.00
00198542-de5a-4292-a500-55ced0b409b0	2023	Especial	7	15571342.32	264712.81944	2120.60	0.00
001c184d-eb08-49c3-a605-2e43042ffad4	2023	Especial	7	13434490.71	228386.34207	2120.60	0.00
001e79fc-650d-4cf8-adfa-b7acf7553b51	2023	Oficial	4	14454384.30	144543.84300	2120.60	0.00
0022a599-fbbb-4570-92b3-e78cac0082d7	2024	Industrial	7	16259444.81	276410.56177	2353.25	2002145.10
0026551c-3347-4cd1-b8ac-926d1a9240ef	2024	Industrial	7	13172943.82	223940.04494	2353.25	2002145.10
0028d703-a114-44e3-bf4b-e1dce3e47e19	2024	Industrial	7	16611313.51	282392.32967	2353.25	2002145.10

- d) Calcular la tasa de impuestos al consumo teniendo en cuenta:
- Tarifa sobre el consumo para calcular la tasa: tarifa_por_destino.csv
 - Cálculo de la tasa: Tarifa sobre consumo * Consumo = Tasa Calculada
 - Valores mínimos aceptados: minimos.csv

iv. Valores máximos aceptados: maximos.csv

Los 4 literales se realizan en la función `process_and_insert_batches`, en el archivo principal `main.py`, previo procesamiento de los .csv en `proces_csv.py` en la siguiente línea.

```
# Fusiones y cálculos máximos y mínimos
batch["tasa_calculada"] = batch["tarifa"] * batch["consumo"]
batch = pd.merge(batch, minimos, on="year", how="left")
batch = pd.merge(batch, maximos, on=["year", "destino"], how="left")

# Ajustar tasa_calculada según mínimos y máximos
batch["tasa_calculada"] = batch["tasa_calculada"].clip(batch["minimo"], batch["maximo"])
```

v. La Tasa Calculada por registro debe modificarse según las reglas de las tablas de mínimos y máximos.

Tomando una muestra, los siguientes registros son modificados con las reglas de mínimos, donde residencial estrato 1 tiene una tarifa de consumo de 0%, la tasa calculada operando da como resultado cero por tanto para 2024 residencial estrato 1 se establece tasa_calculada = mínimo = 2353.25, para 2023 en 2120.6, en máximos no se presenta ningún caso.

	id	year	destino	estrato	consumo	tasa_calculada	minimo	maximo
▶	002e6447-c097-4b71-ab65-b41a7035d367	2024	Residencial	1	9976025.24	2353.25000	2353.25	0.00
	00c0ef03-2cb4-4e53-8f09-b9fca2fcf9b	2024	Residencial	1	17341825.33	2353.25000	2353.25	0.00
	00c55ea4-635a-4b9f-b8e6-9adf0c7c2cce	2024	Residencial	1	13996104.12	2353.25000	2353.25	0.00
	01034ca1-195c-4cc4-a904-1f7b89e4795c	2024	Residencial	1	3729089.72	2353.25000	2353.25	0.00
	02027ed2-5d92-4d0d-b51c-c17f89b96ce8	2024	Residencial	1	17153995.56	2353.25000	2353.25	0.00
	02a50149-6273-49ef-8872-a1c469a826ad	2023	Residencial	1	15714285.02	2120.60000	2120.60	0.00

- e) A medida que los datos son cargados, realiza un seguimiento de:
- Recuento (COUNT) de filas cargadas a la base de datos.
 - Sumatoria (SUM) de la tasa calculada.

NOTA: Se espera que, en la ejecución de este pipeline, al menos después de que se cargue cada .TXT (pero idealmente después de la inserción de cada fila), estas estadísticas se actualicen para reflejar los nuevos datos. Las actualizaciones del Pipeline NO deben tocar los datos ya cargados, es decir, hacer una consulta para traer los valores esperados para cada actualización no es una buena solución al problema.

Es posible observar estos dos parámetros en la terminal de vscode, al ejecutar el pipeline. Adicional la carga por lotes de 500 registros asegura que las estadísticas de filas y tasa calculada se actualicen. Como se realiza con variables acumuladoras en ningún momento el diseño realiza una consulta a la base.

```
# Variables acumuladoras para el seguimiento
total_rows = 0
total_tasa_calculada = 0.0
```

```
# Actualizar estadísticas dinámicas luego de cada microbatch de 500 filas
total_rows += len(batch)
total_tasa_calculada += batch["tasa_calculada"].sum()
```

```
# Imprimir seguimiento del progreso
print(f"Filas cargadas hasta ahora: {total_rows}")
print(f"Sumatoria de tasa calculada hasta ahora: {total_tasa_calculada:.2f}")
```

```
PS C:\Users\angel\OneDrive\Prueba_Tecnica> & C:/Users/angel/AppData/Local/Programs/Python/Python313/python.exe c:/Users/angel/OneDrive/Prueba_Tecnica/main.py
Filas cargadas hasta ahora: 500
Sumatoria de tasa calculada hasta ahora: 67624329.27
Filas cargadas hasta ahora: 1000
Sumatoria de tasa calculada hasta ahora: 142783064.46
Filas cargadas hasta ahora: 1500
Sumatoria de tasa calculada hasta ahora: 217698594.89
Filas cargadas hasta ahora: 2000
```

Comprobante de resultados:

1. Imprime el valor actual de la tasa en ejecución.
2. Realiza una consulta en la base de datos de: recuento total de las, valor promedio, sumatoria, valor mínimo y valor máximo para el campo calculado “Tasa”.

Para estos dos puntos se observa en la terminal de vscode, la tasa de ejecución en los lotes de 500 registros que no hacen llamado alguno a la base y las estadísticas solicitadas estas si se obtienen mediante consulta a la base mediante el llamado de la función `get_statistics` declarada en el archivo `utils.py` e importado al `main.py`. Usando la conexión a la base *Jikkosoft* en MySQL. Es importante concluir que, la sumatoria de tasa calculada hasta ahora: 2669191325.71 obtenida de las variables en el pipeline es igual a la sumatoria de tasa calculada: 2669191325.71 obtenida de una consulta a la base de datos, indicando que se cumple la comprobación.

```
Filas cargadas hasta ahora: 18000
Sumatoria de tasa calculada hasta ahora: 2596197499.34
Filas cargadas hasta ahora: 18500
Sumatoria de tasa calculada hasta ahora: 2669191325.71
Estadísticas finales:
Total filas: 18500
Promedio de tasa calculada: 144280.61
Sumatoria de tasa calculada: 2669191325.71
Tasa mínima: 2120.60
Tasa máxima: 399994.84
PS C:\Users\angel\OneDrive\Prueba_Tecnica> □
```

```

import mysql.connector

def get_statistics(connection):
    cursor = connection.cursor()
    query = """
        SELECT COUNT(*) AS total_rows,
               AVG(tasa_calculada) AS avg_tasa,
               SUM(tasa_calculada) AS sum_tasa,
               MIN(tasa_calculada) AS min_tasa,
               MAX(tasa_calculada) AS max_tasa
        FROM consumos;
    """

```

Algunas reglas y consideraciones del reto:

- Puedes utilizar cualquier Framework o librería que deseas.

Se utiliza `mysql.connector - pandas - os`, en Python.

- Puedes utilizar cualquier base de datos que deseas, bien sea SQL o NoSQL, lo importante es que muestres cómo te conectas a ella, cómo poblas la(s) tabla(s) y como realizas las consultas.

Se utiliza MySQL, se define la función `create_connection` en el archivo `database.py`, esta función se utiliza en la función `get-statistics` definida en el archivo `utils.py`, donde realizamos la consulta de estadísticas directamente a la base `Jikkosoft`, de igual manera se usa la conexión (`connection`) en el archivo principal `main.py` para poblar la base de datos. Es importante aclarar que desde vscode se realiza el query para crear la base de datos en MySQL.

```

01_database.py > ⚒ create_connection
1   # Función para La conexión a MySQL.
2
3   import mysql.connector
4
5   def create_connection():
6       return mysql.connector.connect(
7           host="localhost",
8           user="root",
9           password="MariaPaula18*",
10          database="jikkosoft"
11      )

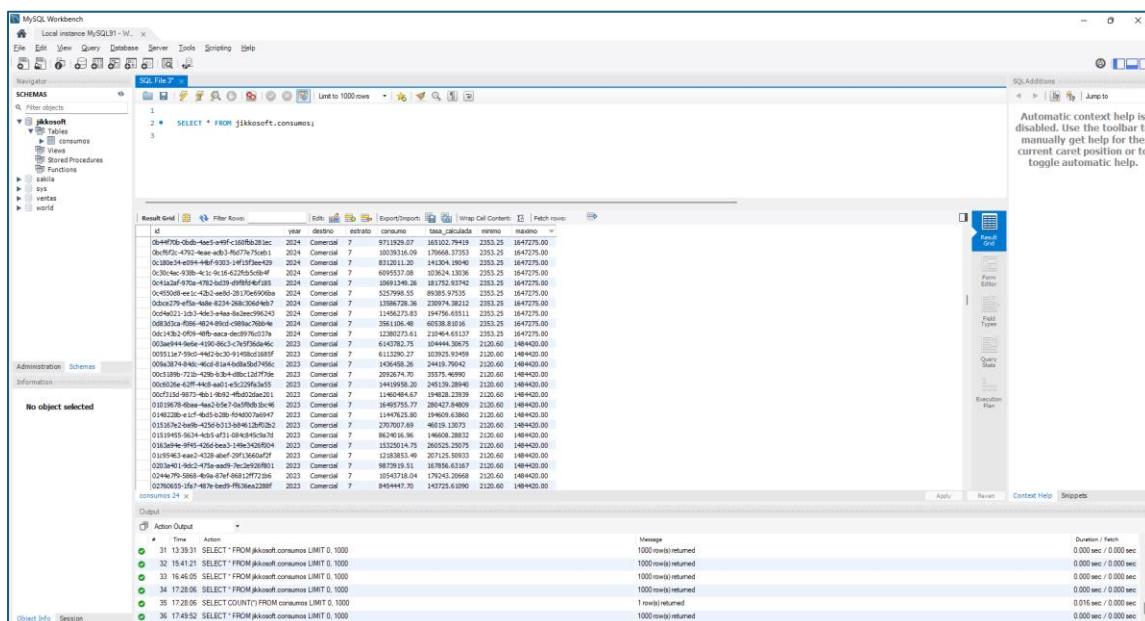
```

```
01_database.py          02_process_txt.py        03_process_csv.py      04_utils.py      05_main.py 5 × jikko
05_main.py > ⚙ process_and_insert_batches
..
62 if __name__ == "__main__":
63     folder_path = r"C:/Users/angel/OneDrive/Prueba_Tecnica/02_dataset"
64     maximos_path = r"C:/Users/angel/OneDrive/Prueba_Tecnica/maximos.csv"
65     minimos_path = r"C:/Users/angel/OneDrive/Prueba_Tecnica/minimos.csv"
66     tarifas_path = r"C:/Users/angel/OneDrive/Prueba_Tecnica/tarifa_por_destino.csv"
67
68     connection = create_connection()
69
70     try:
71         maximos = read_csv_maximos(maximos_path)
72         minimos = read_csv_minimos(minimos_path)
73         tarifas = read_csv_tarifas(tarifas_path)
```

```
C: > Users > angel > .dbclient > query > 1736197332317@@127.0.0.1@3306@jikkosoft > jikkosoft.sql > ...
    ▷ Run | Select | Ask Copilot
1 CREATE TABLE IF NOT EXISTS consumos (
2     id VARCHAR(50) PRIMARY KEY,
3     year INT NOT NULL,
4     destino VARCHAR(50) NOT NULL,
5     estrato INT,
6     consumo FLOAT NOT NULL,
7     tasa_calculada FLOAT,
8     minimo FLOAT NOT NULL,
9     maximo FLOAT NOT NULL
10 );
```

- Puedes hacer uso de alguna interfaz gráfica para administrar/manipular tu base de datos (ejemplo PgAdmin), o puedes hacer uso de línea de comandos.

Se utilizo workbench de MySQL, para confirmar la carga de la base.



- Puedes usar cualquier código existente que tengas a disposición.
 - No hay una forma definida de resolver esta tarea, queremos ver la forma en la que piensas para resolver un problema así.
 - Las estadísticas se pueden almacenar de la forma que deseas: en base de datos, en memoria, en un archivo. Se almacenan en un estadisticas.txt
 - El objetivo es una solución funcional, sin embargo, le damos alto valor al rendimiento.

Acceda a los archivos dando clic en el icono de carpeta.



Orden de ejecución.

1. Crear base de datos jikkosoft, crear tabla consumos, ajuste a decimales.

```
1    ▷ Run
2    CREATE DATABASE jikkosoft;
3
4    ▷ Run | ▷ Select | Ask Copilot
5    CREATE TABLE IF NOT EXISTS consumos (
6        id VARCHAR(50) PRIMARY KEY,
7        year INT NOT NULL,
8        destino VARCHAR(50) NOT NULL,
9        estrato INT,
10       consumo FLOAT NOT NULL,
11       tasa_calculada FLOAT,
12       minimo FLOAT NOT NULL,
13       maximo FLOAT NOT NULL
14   );
15
16   ▷ Run | ▷ Select | Ask Copilot
17   ALTER TABLE consumos
18     MODIFY COLUMN consumo DECIMAL(18, 2),
19     MODIFY COLUMN tasa_calculada DECIMAL(18, 5),
20     MODIFY COLUMN minimo DECIMAL(10, 2),
21     MODIFY COLUMN maximo DECIMAL(10, 2);
```

2. Modifique sus credenciales, en el archivo *database.py*.

```
database.py > ⚡ create_connection
1   # Función para la conexión a MySQL.
2
3   import mysql.connector
4
5   def create_connection():
6       return mysql.connector.connect(
7           host="localhost",
8           user="root",
9           password="MariaPaula18*",
10          database="jikkosoft"
11      )
```

3. Ejecute *main.py*

```
database.py X      process_txt.py      process_csv.py      utils.py      main.py X      jikkosoft.sql ●  
main.py > ...  
1 # Archivo principal que coordina todo.  
2  
3 import pandas as pd  
4 from database import create_connection  
5 from process_txt import read_txt_in_batches  
6 from process_txt import fill_missing_values  
7 from process_csv import read_csv_maximos, read_csv_minimos, read_csv_tarifas  
8 from utils import get_statistics  
9  
10 # Variables acumuladoras para el seguimiento
```

4. Revise los resultados en la terminal de vscode.

```
Filas cargadas hasta ahora: 18000  
Sumatoria de tasa calculada hasta ahora: 2596197499.34  
Filas cargadas hasta ahora: 18500  
Sumatoria de tasa calculada hasta ahora: 2669191325.71  
Estadísticas finales:  
Total filas: 18500  
Promedio de tasa calculada: 144280.61  
Sumatoria de tasa calculada: 2669191325.71  
Tasa mínima: 2120.60  
Tasa máxima: 399994.84
```

Puede ver un video de la ejecución en mi entorno dando clic en el icono.



Altamente agradecido, por la experiencia adquirida en esta prueba técnica.

Cordialmente.

Angel Vera

Ingeniero Electrónico – Magister en Transformación Digital