

Prueba de Ingeniería de Datos – PRAGMA

Descripción General

Este proyecto resuelve una prueba técnica de ingeniería de datos, cuyo objetivo es construir un pipeline de datos que procese archivos CSV en microbatches, almacene los datos en una base de datos relacional y mantenga estadísticas acumuladas en tiempo real, sin recalcular sobre toda la data ya cargada.

Estructura del Proyecto

```
├── data/                # Archivos CSV de entrada
├── src/
│   ├── main.py          # Script principal de orquestación
│   ├── config/          # Configuración de la base de datos
│   ├── etl/             # Lógica de la ETL (Extracción, Transformación, Carga)
│   └── models/          # Modelos/repositorios de tablas
├── requirements.txt     # Dependencias Python
├── docker-compose.yml   # Configuración de PostgreSQL vía Docker
├── .env                 # Variables de entorno para la conexión a la base de
datos
├── README.md            # Documentación del proyecto
└── .gitignore           # Archivos a ignorar por Git
```

¿Qué hace el pipeline?

- Procesa los archivos CSV uno por uno (nunca todos en memoria).
- Inserta los datos en la tabla `transactions` de PostgreSQL.
- Calcula y actualiza estadísticas acumuladas (count, promedio, min, max de `price`) tras cada fila y lote.
- Guarda la evolución de las estadísticas por lote en la tabla `agg_stats`.
- Permite procesar el archivo `validation.csv` en un segundo momento y muestra el impacto en las estadísticas.
- Imprime la evolución de las estadísticas en consola y el resumen final antes y después de cargar `validation.csv`.

Instalación y Ejecución

1. Clona el repositorio y navega al directorio:

```
git clone https://github.com/jblandon97/Prueba-de-ingenier-a-de-datos---PRAGMA.git
cd "Prueba de ingeniería de datos - PRAGMA"
```

2. Instala las dependencias:

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

3. **Configura las variables de entorno en `.env` (ya provisto) para la conexión a la base de datos. A continuación se muestra un ejemplo:**

```
DB_USER=postgres
DB_PASS=postgres
DB_HOST=localhost
DB_PORT=5432
DB_NAME=mydatabase
```

4. **Levanta la base de datos PostgreSQL con Docker:**

```
docker-compose up -d
```

5. **Ejecuta el pipeline para cargar los archivos principales (sin `validation.csv`):**

```
python src/main.py
```

6. **Para procesar `validation.csv` en un segundo momento:**

```
python src/main.py --ejecutar-validation
```

Esquema de la Base de Datos

Tabla: `transactions`

Columna	Tipo	Descripción
id	SERIAL PK	Identificador único de la transacción
timestamp	TIMESTAMP	Fecha y hora del evento
user_id	TEXT	Identificador del usuario
price	FLOAT	Valor de la transacción

Tabla: `agg_stats`

Columna	Tipo	Descripción
---------	------	-------------

Columna	Tipo	Descripción
load_batch	TEXT PK	Nombre del lote procesado (ej: "2012-1.csv")
cum_count	INTEGER	Recuento acumulado de filas hasta ese lote
cum_avg	FLOAT	Promedio acumulado de price
cum_min	FLOAT	Valor mínimo acumulado de price
cum_max	FLOAT	Valor máximo acumulado de price
last_updated	TIMESTAMP	Fecha y hora de la actualización de estadísticas

Tabla: ingested_files

Columna	Tipo	Descripción
batch_name	TEXT PK	Nombre del archivo/lote procesado
loaded_at	TIMESTAMP	Fecha y hora de la carga

Comprobación de Resultados

Al finalizar la ejecución del pipeline, se debe comprobar:

- 1. **Estadísticas en ejecución:**
Se imprimen en consola tras cada lote y tras cada fila (count, promedio, min, max de price).
- 2. **Consulta en la base de datos:**
 - Recuento total de filas (count)
 - Valor promedio (avg)
 - Valor mínimo (min)
 - Valor máximo (max) sobre la tabla transactions.
- 3. **Carga de validation.csv:**
 - Ejecuta el pipeline con el archivo de validación.
 - Muestra el cambio en las estadísticas en ejecución y en la base de datos.

Ejemplo de Resultados

- Estadísticas tras cada lote antes de cargar validation.csv:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS

PS C:\Users\jonat\OneDrive\Documentos\Prueba de ingeniería de datos - PRAGMA> & "C:/Users/jonat/src/main.py"

• Batches ya procesados: set()

● date=2012-01-10 00:00:00	cum_count=1	cum_avg=50.00	cum_min=50.00	cum_max=50.00
● date=2012-01-11 00:00:00	cum_count=2	cum_avg=68.50	cum_min=50.00	cum_max=87.00
● date=2012-01-12 00:00:00	cum_count=3	cum_avg=67.00	cum_min=50.00	cum_max=87.00
● date=2012-01-13 00:00:00	cum_count=4	cum_avg=55.25	cum_min=20.00	cum_max=87.00
● date=2012-01-14 00:00:00	cum_count=5	cum_avg=47.00	cum_min=14.00	cum_max=87.00
● date=2012-01-15 00:00:00	cum_count=6	cum_avg=55.00	cum_min=14.00	cum_max=95.00
● date=2012-01-16 00:00:00	cum_count=7	cum_avg=60.71	cum_min=14.00	cum_max=95.00
● date=2012-01-17 00:00:00	cum_count=8	cum_avg=60.88	cum_min=14.00	cum_max=95.00
● date=2012-01-18 00:00:00	cum_count=9	cum_avg=59.22	cum_min=14.00	cum_max=95.00
● date=2012-01-19 00:00:00	cum_count=10	cum_avg=63.00	cum_min=14.00	cum_max=97.00
● date=2012-01-20 00:00:00	cum_count=11	cum_avg=60.09	cum_min=14.00	cum_max=97.00
● date=2012-01-21 00:00:00	cum_count=12	cum_avg=60.09	cum_min=14.00	cum_max=97.00
● date=2012-01-22 00:00:00	cum_count=13	cum_avg=58.31	cum_min=14.00	cum_max=97.00
● date=2012-01-23 00:00:00	cum_count=14	cum_avg=59.44	cum_min=14.00	cum_max=97.00
● date=2012-01-24 00:00:00	cum_count=15	cum_avg=58.47	cum_min=14.00	cum_max=97.00
● date=2012-01-25 00:00:00	cum_count=16	cum_avg=56.51	cum_min=14.00	cum_max=97.00
● date=2012-01-26 00:00:00	cum_count=17	cum_avg=57.77	cum_min=14.00	cum_max=97.00
● date=2012-01-27 00:00:00	cum_count=18	cum_avg=56.89	cum_min=14.00	cum_max=97.00
● date=2012-01-28 00:00:00	cum_count=19	cum_avg=56.27	cum_min=14.00	cum_max=97.00
● date=2012-01-29 00:00:00	cum_count=20	cum_avg=58.25	cum_min=14.00	cum_max=97.00
● date=2012-01-30 00:00:00	cum_count=21	cum_avg=58.25	cum_min=14.00	cum_max=97.00
● date=2012-01-31 00:00:00	cum_count=22	cum_avg=59.61	cum_min=14.00	cum_max=97.00
● batch=2012-1	cum_count=22	cum_avg=59.61	cum_min=14.00	cum_max=97.00
● date=2012-02-01 00:00:00	cum_count=23	cum_avg=60.32	cum_min=14.00	cum_max=97.00
● date=2012-02-02 00:00:00	cum_count=24	cum_avg=60.56	cum_min=14.00	cum_max=97.00
● date=2012-02-03 00:00:00	cum_count=25	cum_avg=60.01	cum_min=14.00	cum_max=97.00
● date=2012-02-04 00:00:00	cum_count=26	cum_avg=61.55	cum_min=14.00	cum_max=100.00
● date=2012-02-05 00:00:00	cum_count=27	cum_avg=59.98	cum_min=14.00	cum_max=100.00
● date=2012-02-06 00:00:00	cum_count=28	cum_avg=59.94	cum_min=14.00	cum_max=100.00
● date=2012-02-07 00:00:00	cum_count=29	cum_avg=60.53	cum_min=14.00	cum_max=100.00
● date=2012-02-08 00:00:00	cum_count=30	cum_avg=61.84	cum_min=14.00	cum_max=100.00
● date=2012-02-09 00:00:00	cum_count=31	cum_avg=61.62	cum_min=14.00	cum_max=100.00
● date=2012-02-10 00:00:00	cum_count=32	cum_avg=60.29	cum_min=14.00	cum_max=100.00
● date=2012-02-11 00:00:00	cum_count=33	cum_avg=61.34	cum_min=14.00	cum_max=100.00
● date=2012-02-12 00:00:00	cum_count=34	cum_avg=61.69	cum_min=14.00	cum_max=100.00
● date=2012-02-13 00:00:00	cum_count=35	cum_avg=60.84	cum_min=14.00	cum_max=100.00
● date=2012-02-14 00:00:00	cum_count=36	cum_avg=61.79	cum_min=14.00	cum_max=100.00
● date=2012-02-15 00:00:00	cum_count=37	cum_avg=60.44	cum_min=12.00	cum_max=100.00
● date=2012-02-16 00:00:00	cum_count=38	cum_avg=59.82	cum_min=12.00	cum_max=100.00
● date=2012-02-17 00:00:00	cum_count=39	cum_avg=59.01	cum_min=12.00	cum_max=100.00
● date=2012-02-18 00:00:00	cum_count=40	cum_avg=57.78	cum_min=10.00	cum_max=100.00
● date=2012-02-19 00:00:00	cum_count=41	cum_avg=57.03	cum_min=10.00	cum_max=100.00
● date=2012-02-20 00:00:00	cum_count=42	cum_avg=56.01	cum_min=10.00	cum_max=100.00
● date=2012-02-21 00:00:00	cum_count=43	cum_avg=56.73	cum_min=10.00	cum_max=100.00
● date=2012-02-22 00:00:00	cum_count=44	cum_avg=57.14	cum_min=10.00	cum_max=100.00

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS	POSTGRESQL QUERY RESULTS
●	date=2012-04-23 00:00:00		cum_count=105		cum_avg=57.86 cum_min=10.00 cum_max=100.00
●	date=2012-04-24 00:00:00		cum_count=106		cum_avg=58.08 cum_min=10.00 cum_max=100.00
●	date=2012-04-25 00:00:00		cum_count=107		cum_avg=57.94 cum_min=10.00 cum_max=100.00
●	date=2012-04-26 00:00:00		cum_count=108		cum_avg=57.92 cum_min=10.00 cum_max=100.00
●	date=2012-04-27 00:00:00		cum_count=109		cum_avg=57.92 cum_min=10.00 cum_max=100.00
●	date=2012-04-28 00:00:00		cum_count=110		cum_avg=57.87 cum_min=10.00 cum_max=100.00
●	date=2012-04-29 00:00:00		cum_count=111		cum_avg=57.65 cum_min=10.00 cum_max=100.00
●	date=2012-04-30 00:00:00		cum_count=112		cum_avg=57.81 cum_min=10.00 cum_max=100.00
●	batch=2012-4		cum_count=112		cum_avg=57.81 cum_min=10.00 cum_max=100.00
●	date=2012-05-01 00:00:00		cum_count=113		cum_avg=58.03 cum_min=10.00 cum_max=100.00
●	date=2012-05-02 00:00:00		cum_count=114		cum_avg=58.05 cum_min=10.00 cum_max=100.00
●	date=2012-05-03 00:00:00		cum_count=115		cum_avg=57.71 cum_min=10.00 cum_max=100.00
●	date=2012-05-04 00:00:00		cum_count=116		cum_avg=58.02 cum_min=10.00 cum_max=100.00
●	date=2012-05-05 00:00:00		cum_count=117		cum_avg=57.67 cum_min=10.00 cum_max=100.00
●	date=2012-05-06 00:00:00		cum_count=118		cum_avg=57.35 cum_min=10.00 cum_max=100.00
●	date=2012-05-07 00:00:00		cum_count=119		cum_avg=57.42 cum_min=10.00 cum_max=100.00
●	date=2012-05-08 00:00:00		cum_count=120		cum_avg=57.05 cum_min=10.00 cum_max=100.00
●	date=2012-05-09 00:00:00		cum_count=121		cum_avg=56.83 cum_min=10.00 cum_max=100.00
●	date=2012-05-10 00:00:00		cum_count=122		cum_avg=57.15 cum_min=10.00 cum_max=100.00
●	date=2012-05-11 00:00:00		cum_count=123		cum_avg=57.31 cum_min=10.00 cum_max=100.00
●	date=2012-05-12 00:00:00		cum_count=124		cum_avg=56.97 cum_min=10.00 cum_max=100.00
●	date=2012-05-13 00:00:00		cum_count=125		cum_avg=56.69 cum_min=10.00 cum_max=100.00
●	date=2012-05-14 00:00:00		cum_count=126		cum_avg=56.97 cum_min=10.00 cum_max=100.00
●	date=2012-05-15 00:00:00		cum_count=127		cum_avg=56.87 cum_min=10.00 cum_max=100.00
●	date=2012-05-16 00:00:00		cum_count=128		cum_avg=57.08 cum_min=10.00 cum_max=100.00
●	date=2012-05-17 00:00:00		cum_count=129		cum_avg=57.13 cum_min=10.00 cum_max=100.00
●	date=2012-05-18 00:00:00		cum_count=130		cum_avg=57.33 cum_min=10.00 cum_max=100.00
●	date=2012-05-19 00:00:00		cum_count=131		cum_avg=57.12 cum_min=10.00 cum_max=100.00
●	date=2012-05-20 00:00:00		cum_count=132		cum_avg=57.16 cum_min=10.00 cum_max=100.00
●	date=2012-05-21 00:00:00		cum_count=133		cum_avg=57.40 cum_min=10.00 cum_max=100.00
●	date=2012-05-21 00:00:00		cum_count=133		cum_avg=57.40 cum_min=10.00 cum_max=100.00
●	date=2012-05-22 00:00:00		cum_count=134		cum_avg=57.63 cum_min=10.00 cum_max=100.00
●	date=2012-05-23 00:00:00		cum_count=135		cum_avg=57.45 cum_min=10.00 cum_max=100.00
●	date=2012-05-22 00:00:00		cum_count=134		cum_avg=57.63 cum_min=10.00 cum_max=100.00
●	date=2012-05-23 00:00:00		cum_count=135		cum_avg=57.45 cum_min=10.00 cum_max=100.00
●	date=2012-05-23 00:00:00		cum_count=135		cum_avg=57.45 cum_min=10.00 cum_max=100.00
●	date=2012-05-23 00:00:00		cum_count=135		cum_avg=57.45 cum_min=10.00 cum_max=100.00
●	date=2012-05-24 00:00:00		cum_count=136		cum_avg=57.76 cum_min=10.00 cum_max=100.00
●	date=2012-05-25 00:00:00		cum_count=137		cum_avg=57.90 cum_min=10.00 cum_max=100.00
●	date=2012-05-26 00:00:00		cum_count=138		cum_avg=58.14 cum_min=10.00 cum_max=100.00
●	date=2012-05-27 00:00:00		cum_count=139		cum_avg=58.16 cum_min=10.00 cum_max=100.00
●	date=2012-05-28 00:00:00		cum_count=140		cum_avg=58.23 cum_min=10.00 cum_max=100.00
●	date=2012-05-29 00:00:00		cum_count=141		cum_avg=58.09 cum_min=10.00 cum_max=100.00
●	date=2012-05-30 00:00:00		cum_count=142		cum_avg=58.08 cum_min=10.00 cum_max=100.00
●	date=2012-05-31 00:00:00		cum_count=143		cum_avg=57.91 cum_min=10.00 cum_max=100.00
●	batch=2012-5		cum_count=143		cum_avg=57.91 cum_min=10.00 cum_max=100.00
=== Estadísticas antes de validation.csv ===					
== Resumen de las estadísticas ==					
Total filas: 143					
Promedio price: 57.91					
Mínimo price: 10.0					
Máximo price: 100.0					

- Consulta de las tablas `transactions` y `agg_stats` antes de cargar `validation.csv`:

The screenshot shows a PostgreSQL IDE with a query window open. The query window contains two SQL queries. The first query selects from the public.transactions table, and the second query selects from the public.agg_stats table. The results of the queries are displayed in two tables.

Query 1: SELECT id, timestamp, user_id, price FROM public.transactions LIMIT 1000;

id	timestamp	user_id	price
1	2012-01-10 00:00:00	9	50.0
2	2012-01-11 00:00:00	10	87.0
3	2012-01-12 00:00:00	7	64.0
4	2012-01-13 00:00:00	10	20.0
5	2012-01-14 00:00:00	10	14.0
6	2012-01-15 00:00:00	8	95.0
7	2012-01-16 00:00:00	2	95.0
8	2012-01-17 00:00:00	4	62.0
9	2012-01-18 00:00:00	8	46.0
10	2012-01-19 00:00:00	2	97.0

Query 2: SELECT load_batch, cum_count, cum_avg, cum_min, cum_max, last_updated FROM public.agg_stats LIMIT 1000;

load_batch	cum_count	cum_avg	cum_min	cum_max	last_updated
2012-3	82	57.94323725055432	10.0	100.0	2025-06-01T06:08:25.996016
2012-1	22	59.60661157024793	14.0	97.0	2025-06-01T06:08:25.601571
2012-4	112	57.81122263389483	10.0	100.0	2025-06-01T06:08:26.175481
2012-2	51	56.88912655971479	10.0	100.0	2025-06-01T06:08:25.793552
2012-5	143	57.908090454519034	10.0	100.0	2025-06-01T06:08:26.366932

- Estadísticas después de cargar **validation.csv**:

```
PS C:\Users\jonat\OneDrive\Documentos\Prueba de ingeniería de datos - PRAGMA> python src/main.py --ejecutar-validation
Batches ya procesados: {'2012-4', '2012-1', '2012-3', '2012-5', '2012-2'}

=== Ejecutando validation.csv ===
● date=2012-06-01 00:00:00 | cum_count=144 | cum_avg=57.69 | cum_min=10.00 | cum_max=100.00
● date=2012-06-02 00:00:00 | cum_count=145 | cum_avg=57.36 | cum_min=10.00 | cum_max=100.00
● date=2012-06-03 00:00:00 | cum_count=146 | cum_avg=57.06 | cum_min=10.00 | cum_max=100.00
● date=2012-06-04 00:00:00 | cum_count=147 | cum_avg=57.30 | cum_min=10.00 | cum_max=100.00
● date=2012-06-05 00:00:00 | cum_count=148 | cum_avg=57.12 | cum_min=10.00 | cum_max=100.00
● date=2012-06-06 00:00:00 | cum_count=149 | cum_avg=57.15 | cum_min=10.00 | cum_max=100.00
● date=2012-06-07 00:00:00 | cum_count=150 | cum_avg=56.86 | cum_min=10.00 | cum_max=100.00
● date=2012-06-08 00:00:00 | cum_count=151 | cum_avg=57.05 | cum_min=10.00 | cum_max=100.00
● batch=validation | cum_count=151 | cum_avg=57.05 | cum_min=10.00 | cum_max=100.00
✓ 'validation' procesado correctamente.
=== Estadísticas después de validation.csv ===
== Resumen de las estadísticas ==
Total filas: 151
Promedio price: 57.05
Mínimo price: 10.0
Máximo price: 100.0
```

- Consulta de las tablas **transactions** y **agg_stats** después de cargar **validation.csv**:

The screenshot shows a PostgreSQL interface with a query window and a results pane. The query window contains two SQL queries. The first query selects columns from the `public.transactions` table, and the second query selects columns from the `public.agg_stats` table. The results pane displays two tables of data.

Query 1 Results:

id	timestamp	user_id	price
142	2012-05-30 00:00:00	8	57.0
143	2012-05-31 00:00:00	6	33.0
144	2012-06-01 00:00:00	10	26.0
145	2012-06-02 00:00:00	8	11.0
146	2012-06-03 00:00:00	9	13.0
147	2012-06-04 00:00:00	7	92.0
148	2012-06-05 00:00:00	10	31.0
149	2012-06-06 00:00:00	3	62.0
150	2012-06-07 00:00:00	4	13.0
151	2012-06-08 00:00:00	7	86.0

Query 2 Results:

load_batch	cum_count	cum_avg	cum_min	cum_max	last_updated	
1	2012-1	22	59.60661157024793	14.0	97.0	2025-06-01T06:08:25.601571
2	2012-2	51	56.88912655971479	10.0	100.0	2025-06-01T06:08:25.793552
3	2012-3	82	57.94323725055432	10.0	100.0	2025-06-01T06:08:25.996016
4	2012-4	112	57.81122263389483	10.0	100.0	2025-06-01T06:08:26.175481
5	2012-5	143	57.908090454519034	10.0	100.0	2025-06-01T06:08:26.366932
6	validation	151	57.052032682094186	10.0	100.0	2025-06-01T06:20:03.852049

Si se vuelve a ejecutar el pipeline, se debe observar que los archivos ya procesados no se vuelven a cargar y las estadísticas no se actualizan, manteniendo la integridad de los datos.

- Consulta de las tablas `transactions` y `agg_stats` tras re-ejecución:

The screenshot shows a terminal window with the following output:

```
PS C:\Users\jonat\OneDrive\Documentos\Prueba de ingeniería de datos - PRAGMA> python src/main.py
Batches ya procesados: {'validation', '2012-5', '2012-3', '2012-2', '2012-1', '2012-4'}

▲ '2012-1' ya procesado, omitiendo.
▲ '2012-2' ya procesado, omitiendo.
▲ '2012-3' ya procesado, omitiendo.
▲ '2012-4' ya procesado, omitiendo.
▲ '2012-5' ya procesado, omitiendo.
=== Estadísticas antes de validation.csv ===
== Resumen de las estadísticas ==
Total filas: 151
Promedio price: 57.05
Mínimo price: 10.0
Máximo price: 100.0

PS C:\Users\jonat\OneDrive\Documentos\Prueba de ingeniería de datos - PRAGMA> python src/main.py --ejecutar-validation
Batches ya procesados: {'2012-1', '2012-4', '2012-5', '2012-2', '2012-3', 'validation'}

▲ 'validation' ya procesado, omitiendo.
=== Estadísticas después de validation.csv ===
== Resumen de las estadísticas ==
Total filas: 151
Promedio price: 57.05
Mínimo price: 10.0
Máximo price: 100.0

PS C:\Users\jonat\OneDrive\Documentos\Prueba de ingeniería de datos - PRAGMA>
```

Notas Técnicas

- Las estadísticas se actualizan de forma incremental, sin recalcularse sobre toda la tabla.
- El diseño permite auditar la evolución de las estadísticas por lote.

- Puedes administrar la base de datos con PgAdmin, la línea de comandos u otra herramienta. En mi caso, utilicé la extensión de PostgreSQL de Microsoft para Visual Studio Code. Deberás configurar la conexión con las mismas variables de entorno que en el archivo `.env`.