NYC Ride-Hailing Dashboard 🚖

Objetivo: Visualizar patrones de viaje Uber, Lyft y más en NYC, 2024.

Dataset

Archivos	.parquet mensuales
Registros	~20 millones por mes
Columnas clave	Fechas, zonas, tarifas, impuestos, operador

🔧 Transformaciones realizadas

Muestreo estratificado

5% por operador para balancear datos

Merge

Taxi_zone_lookup.csv añade pickup_zone

Columnas derivadas

pickup_hour, pickup_weekday, pickup_month

✨ Estandarización de Operadores

Filtra para procesar solo archivos Parquet del año 2024 (basado en el nombre del archivo).

Para cada archivo seleccionado:

- Lo carga en un DataFrame.
- Si aplica, realiza un muestreo estratificado del 5% basado en `hvfhs_license_num`.
- Añade nombres de zonas de recogida uniendo con la tabla de zonas.
- Convierte una columna de fecha/hora y extrae la hora, día de la semana y mes.
- Guarda el DataFrame resultante (más pequeño y enriquecido) en un nuevo archivo Parquet en el directorio de salida.Se unificaron nombres y categorías para Uber, Lyft y otros.

```python
# Recorrer archivos
for file in sorted(os.listdir(input_dir)):
    if not file.endswith(".parquet") or "2024 -" not in file:
        continue

    print(f"🔄 Procesando: {file}")
    path = os.path.join(input_dir, file)
    df = pd.read_parquet(path)

    # Muestreo estratificado
    if "hvfhs_license_num" in df.columns:
        df = df.groupby("hvfhs_license_num", group_keys=False).apply(
            lambda x: x.sample(frac=0.05, random_state=42)
        )

    # Merge con zonas
    if "PULocationID" in df.columns:
        df["PULocationID"] = df["PULocationID"].astype("int32")
        df = df.merge(zones[["PULocationID", "pickup_zone"]], on="PULocationID", how="left")

    # Procesar datetime
    df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"], errors="coerce")
    df["pickup_hour"] = df["pickup_datetime"].dt.hour
    df["pickup_weekday"] = df["pickup_datetime"].dt.weekday
    df["pickup_month"] = df["pickup_datetime"].dt.month

    # Guardar
    output_path = os.path.join(output_dir, file.replace(".parquet", "_reduced.parquet"))
    df.to_parquet(output_path, index=False)
    print(f"✅ Guardado: {output_path} ({len(df):,} filas)")
```

✨ Estandarización de Operadores

HV0003

Uber

HV0005

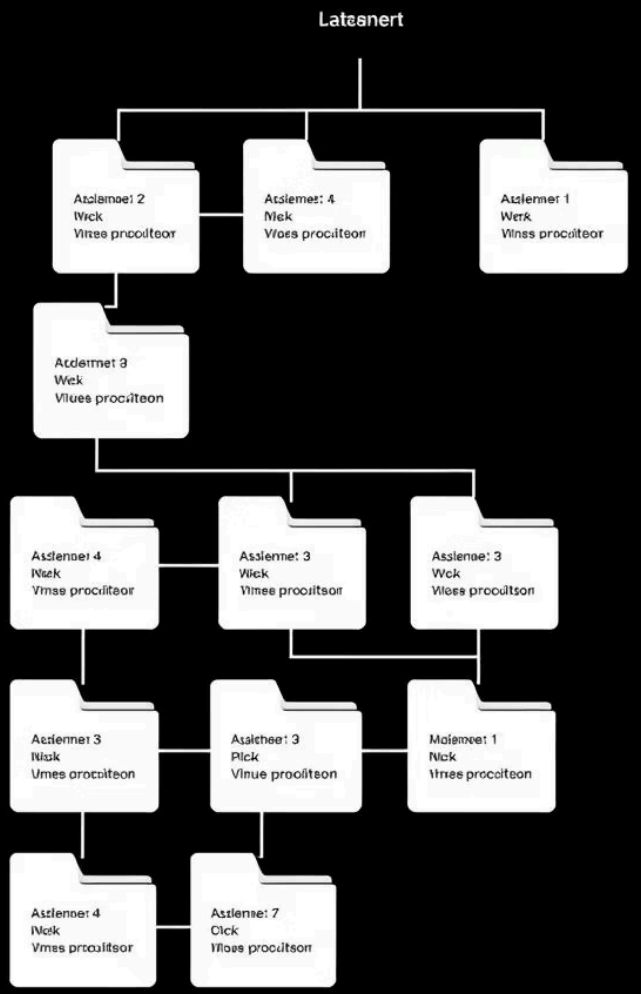
Lyft

HV0002

Via

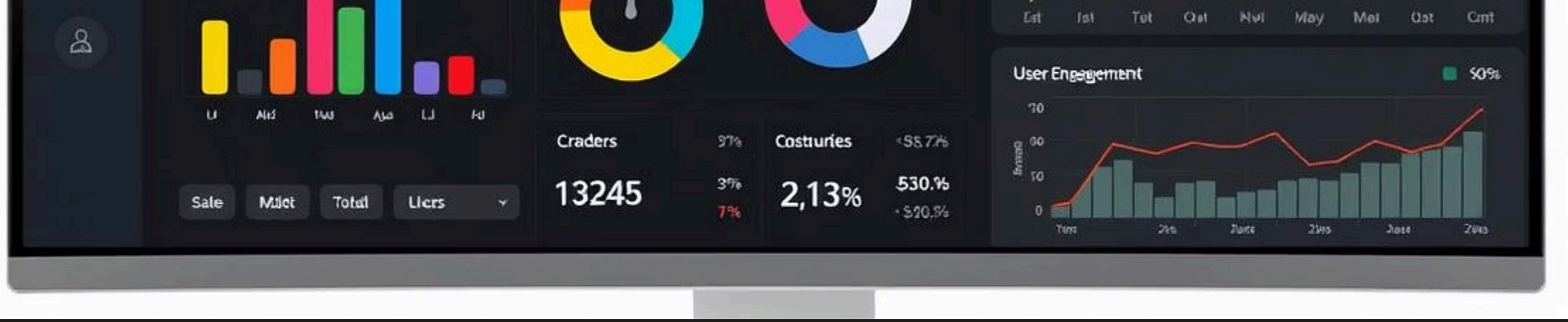
HV0004

Curb



📂 Estructura del Proyecto

```
mineria/
├── raw-data/     #Parquets mensuales originales
├── data_sampled/ #Parquets procesados y reducidos
├── data/         #Lookup de zonas
├── app.py        #Dashboard interactivo
├── generate_reduced_parquets.py
└── map_license_names.py
```



🔍 Dashboard Interactivo

Filtros

- Mes
- Operador
- Hora
- Zona

Visualizaciones

- Histograma por hora
- Día de la semana
- Zonas frecuentes
- Propinas y pagos

🚀 Ejecución y Resultados

Comandos clave

1. conda activate dask-cpu
2. streamlit run app.py
3. Acceso: **http://localhost:8501**

Resultados

- Datos reducidos a ~1 millón (5% estratificado)
- Dashboard fluido y eficiente
- Escalable a mapas y análisis avanzado