S20 - Unidad 2. Evidencia de Aprendizaje (EA2), de la Actividad 2: Despliegue de una Infraestructura virtual.

Andres Felipe Callejas

Yonier Alexis Quiceno Rodriguez

Big Data

IU Digital (Institución Universitaria Digital de Antioquia)

Colombia  
2025

Contenido

[Introducción 2](#_Toc197896041)

[Descripción del problema. 3](#_Toc197896042)

[Objetivos. 4](#_Toc197896043)

[General 4](#_Toc197896044)

[Especifico 4](#_Toc197896045)

[Descripción de los datos disponibles. 4](#_Toc197896046)

[Solución propuesta (Elección del SGBD y esquema diseñado). 5](#_Toc197896047)

[Metodología empleada. 5](#_Toc197896048)

[Resultados y Conclusiones. 6](#_Toc197896049)

[Bibliografía (Normas APA). 7](#_Toc197896050)

# Introducción

En el contexto actual, la gestión y análisis de grandes volúmenes de datos climáticos es esencial para la toma de decisiones empresariales en múltiples sectores. Este informe documenta el proceso completo de despliegue de una infraestructura virtual utilizando \*\*Databricks Community Edition\*\* como plataforma open source, para analizar el comportamiento climático de Medellín, Colombia, durante el periodo 2019-2024.

El proyecto surge de la necesidad de la empresa ClimaSmartde contar con una plataforma escalable y eficiente para el procesamiento de datos meteorológicos históricos. Utilizando datos obtenidos de Kaggle, se implementa una arquitectura cloud-based que permite la ingesta, almacenamiento, procesamiento y visualización de datos climáticos, facilitando la toma de decisiones basadas en evidencia para servicios y productos relacionados con el clima.

La elección de Databricks Community Edition responde a la necesidad de una solución que combine las ventajas del procesamiento distribuido con Apache Spark, la facilidad de uso de notebooks interactivos, y la capacidad de escalar según las necesidades futuras de la empresa, todo esto sin incurrir en costos iniciales de infraestructura.

# Descripción del problema.

La empresa ClimaSmart enfrenta varios desafíos críticos en su operación actual:

1. Ausencia de infraestructura centralizada: Los datos climáticos se encuentran dispersos en diferentes formatos y ubicaciones, dificultando su acceso y análisis integral.

2. Limitaciones de procesamiento: Las herramientas actuales (hojas de cálculo tradicionales) no pueden manejar eficientemente el volumen de datos históricos de 6 años (2019-2024), que comprende más de 2,000 registros diarios.

3. Falta de capacidades analíticas avanzadas: No existe una plataforma que permita realizar análisis predictivos o identificar patrones complejos en los datos meteorológicos.

4. Escalabilidad limitada: La infraestructura actual no puede crecer según las necesidades del negocio ni integrar nuevas fuentes de datos.

5. Colaboración ineficiente: Los equipos de análisis no pueden trabajar simultáneamente sobre los mismos datasets, limitando la productividad.

Esta situación impide que ClimaSmart pueda:

- Ofrecer servicios de predicción climática personalizados

- Identificar tendencias y anomalías climáticas

- Optimizar sus productos basándose en patrones históricos

- Responder rápidamente a consultas de clientes sobre condiciones climáticas específicas

# Objetivos.

## General

Desplegar una infraestructura virtual en Databricks Community Edition para la empresa ClimaSmart, usando datos climáticos de Medellín (2019-2024) provenientes de Kaggle, habilitando así capacidades de análisis y visualización.

## Especifico

* Seleccionar y justificar la herramienta open source de infraestructura virtual.
* Diseñar la arquitectura de la solución, considerando almacenamiento, cómputo y acceso a datos.
* Configurar la infraestructura y cargar el dataset de Kaggle.
* Desplegar y migrar los datos a la plataforma.
* Validar la correcta creación de tablas y visualización de los datos.

# Descripción de los datos disponibles

El dataset utilizado proviene de Kaggle, específicamente del repositorio "historical-weather-medelln" creado por Jaider Lopez. Las características principales del conjunto de datos son:

## Estructura del Dataset

* Periodo temporal: 2019-2024 (6 años completos)
* Registros totales: 2,158 observaciones diarias
* Formato original: CSV con codificación Latin-1
* Tamaño: Aproximadamente 17 KB comprimido

## Calidad de los datos

* Completitud: 100% - No se encontraron valores nulos en ninguna columna
* Consistencia: Alta - Todos los valores están dentro de rangos meteorológicos esperados para Medellín
* Formato: Uniforme - Fechas en formato ISO, unidades métricas estándar

## Estadísticas descriptivas clave

* Temperatura máxima promedio: 22.48°C (±2.84°C)
* Temperatura mínima promedio: 10.89°C (±1.88°C)
* Precipitación promedio: 13.95 mm (±13.74 mm)
* Velocidad del viento promedio: 4.21 km/h (±1.55 km/h)

# ****Solución propuesta****

## Elección de la herramienta

Se selecciona **Databricks Community Edition** por ser una plataforma gratuita, basada en Apache Spark, que ofrece entorno colaborativo para procesamiento de datos, integración con notebooks, y facilidades para ingestión y transformación de datos a escala. Su modelo cloud elimina la necesidad de gestionar hardware local y permite escalabilidad.

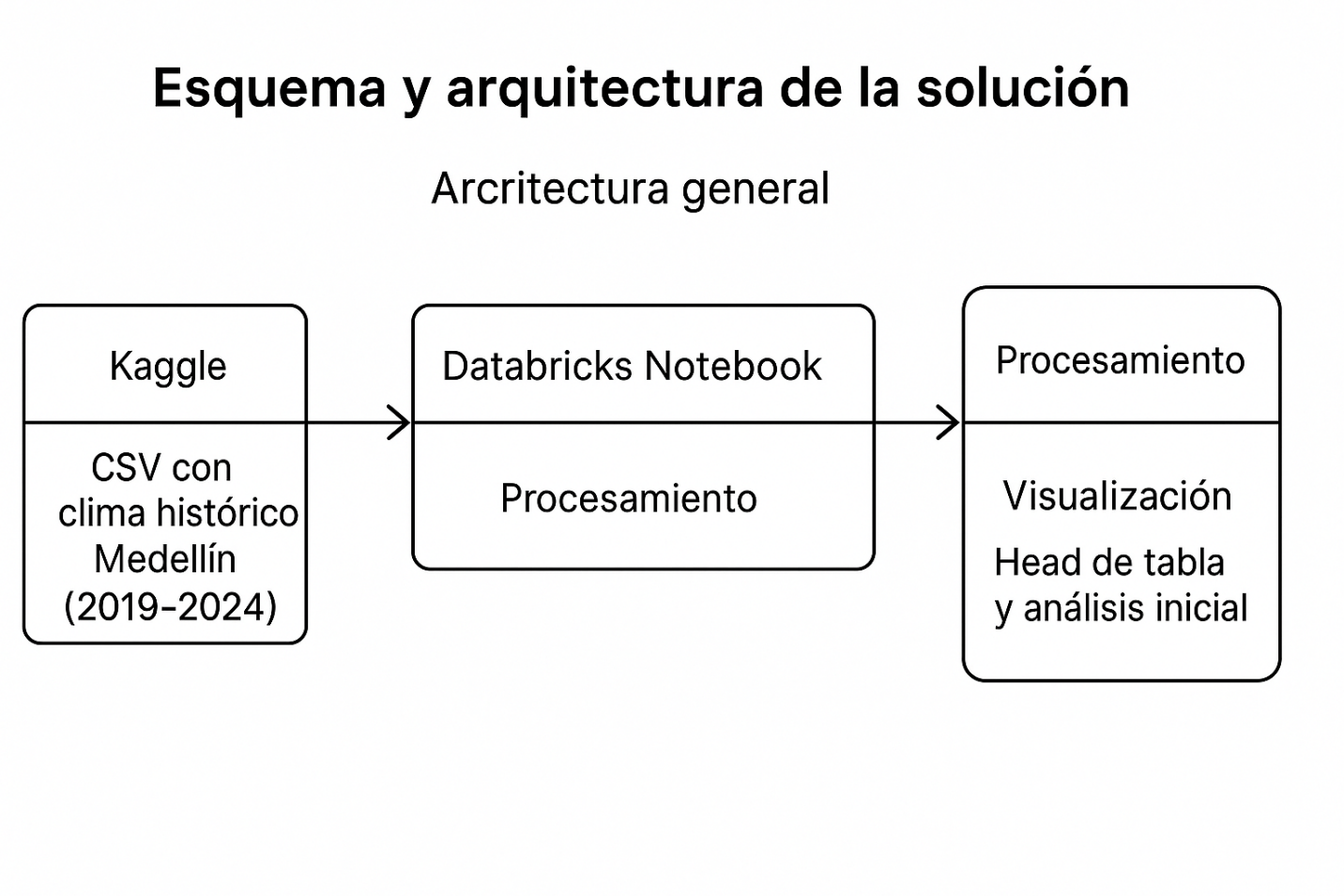
## Esquema y arquitectura de la solución

Arquitectura general:

* Fuente de datos: Kaggle - CSV con clima histórico Medellín (2019-2024).
* Plataforma de cómputo: Databricks Community Edition (basado en Spark).
* Almacenamiento: DBFS (Databricks File System).
* Procesamiento: Notebook Databricks (Python, pandas, PySpark).
* Visualización: Head de tabla y análisis inicial.

## Figura 1

## Diagrama de esquema



## Configuración

* Creación de cuenta en Databricks Community Edition
* Registro en [Databricks Community Edition](https://community.cloud.databricks.com/).
* Configuración de clúster:
* Databricks runtime version:  
  Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

(Nota: se recomienda usar DBR 13.3 LTS o superior si está disponible)

* Versión de Python:

PYSPARK\_PYTHON=/databricks/python3/bin/python3

Indica que se está utilizando Python 3 (ruta estándar de Python 3 en Databricks).

* Instance (Recursos asignados):

Free 15 GB Memory:

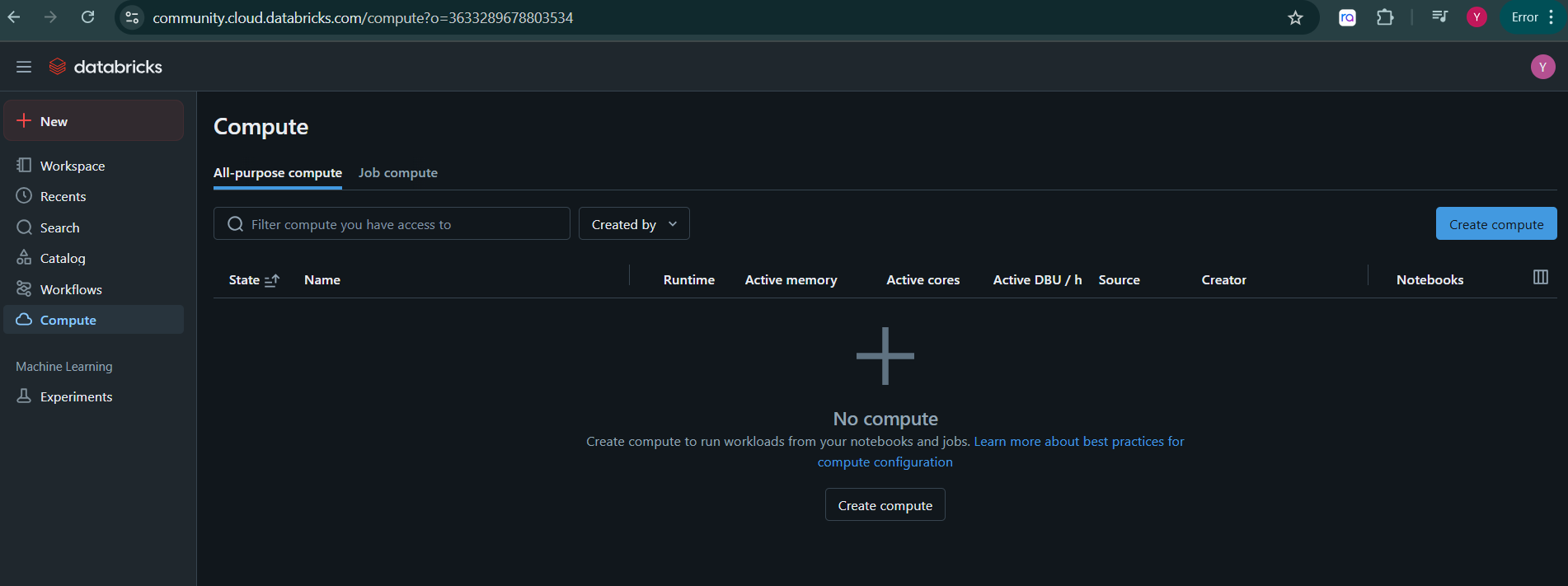
(Como edición Community, la máquina virtual asignada tiene hasta 15 GB de memoria RAM compartida, aunque el uso real depende del sistema.)

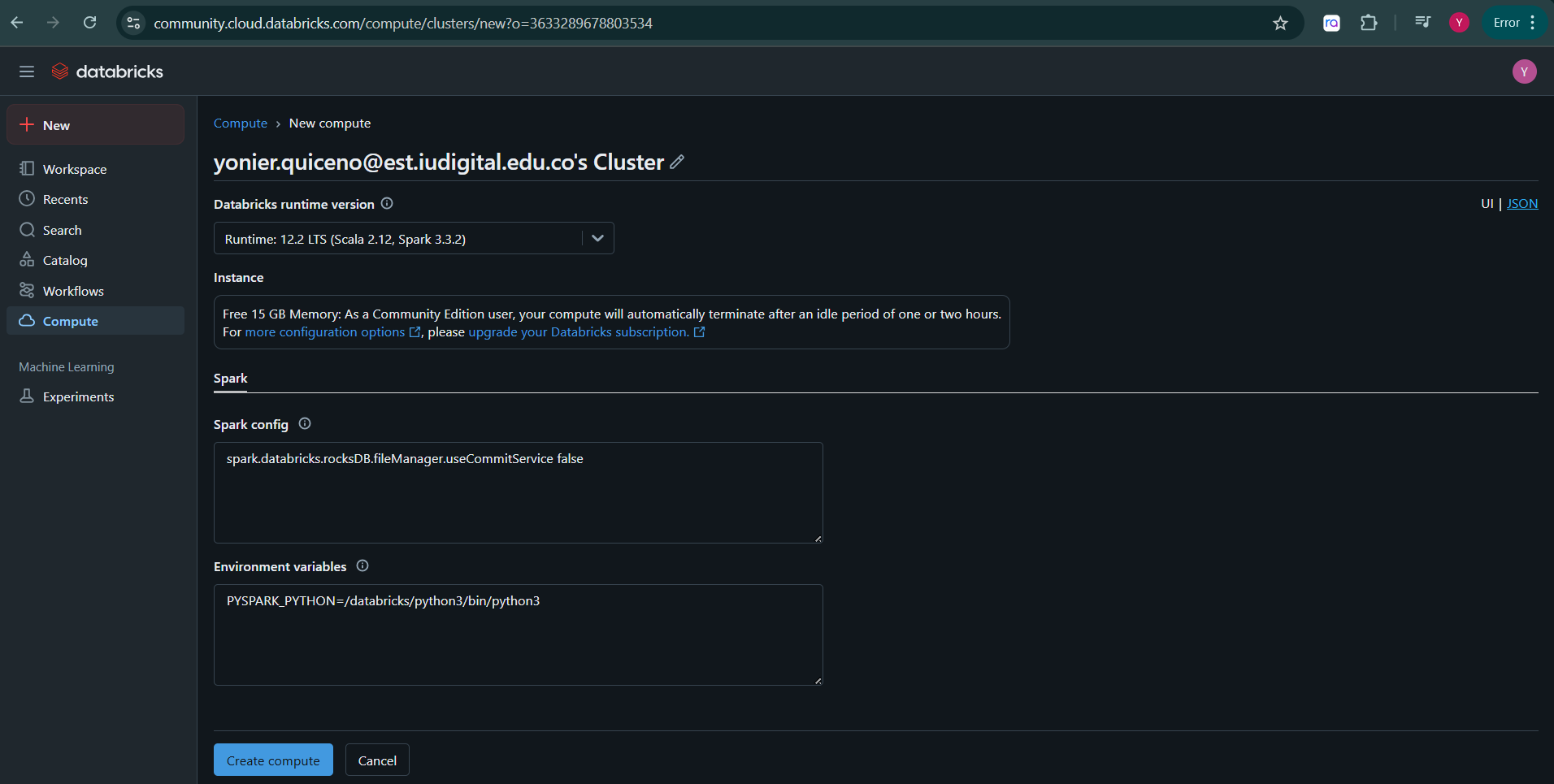
CPU:

1 núcleo (por límite de la edición Community)

### Figura 2

### Creación clúster de cómputo.



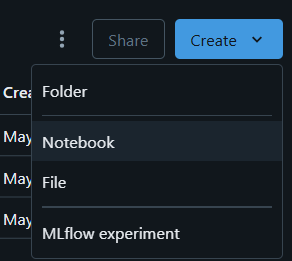


Para el procesamiento y análisis de los datos, se configuró un clúster en Databricks Community Edition utilizando el Databricks Runtime 12.2 LTS (Scala 2.12, Spark 3.3.2), con recursos asignados de hasta 15 GB de RAM compartida y 1 CPU, según las limitaciones de la edición gratuita. El entorno emplea Python 3 como intérprete principal (definido por la variable de entorno PYSPARK\_PYTHON=/databricks/python3/bin/python3), lo que permite el uso de PySpark y pandas para manipulación de datos. Además, se estableció una configuración especial de Spark (spark.databricks.rocksDB.fileManager.useCommitService false) para optimizar el manejo de archivos internos. Es importante tener en cuenta que, por políticas de la edición Community, el clúster se apaga automáticamente tras uno o dos horas de inactividad. Esta configuración asegura un entorno estable y compatible para ejecutar análisis exploratorios y prototipos sobre los datos de clima histórico de Medellín.

* Carga de Notebooks y librerías:

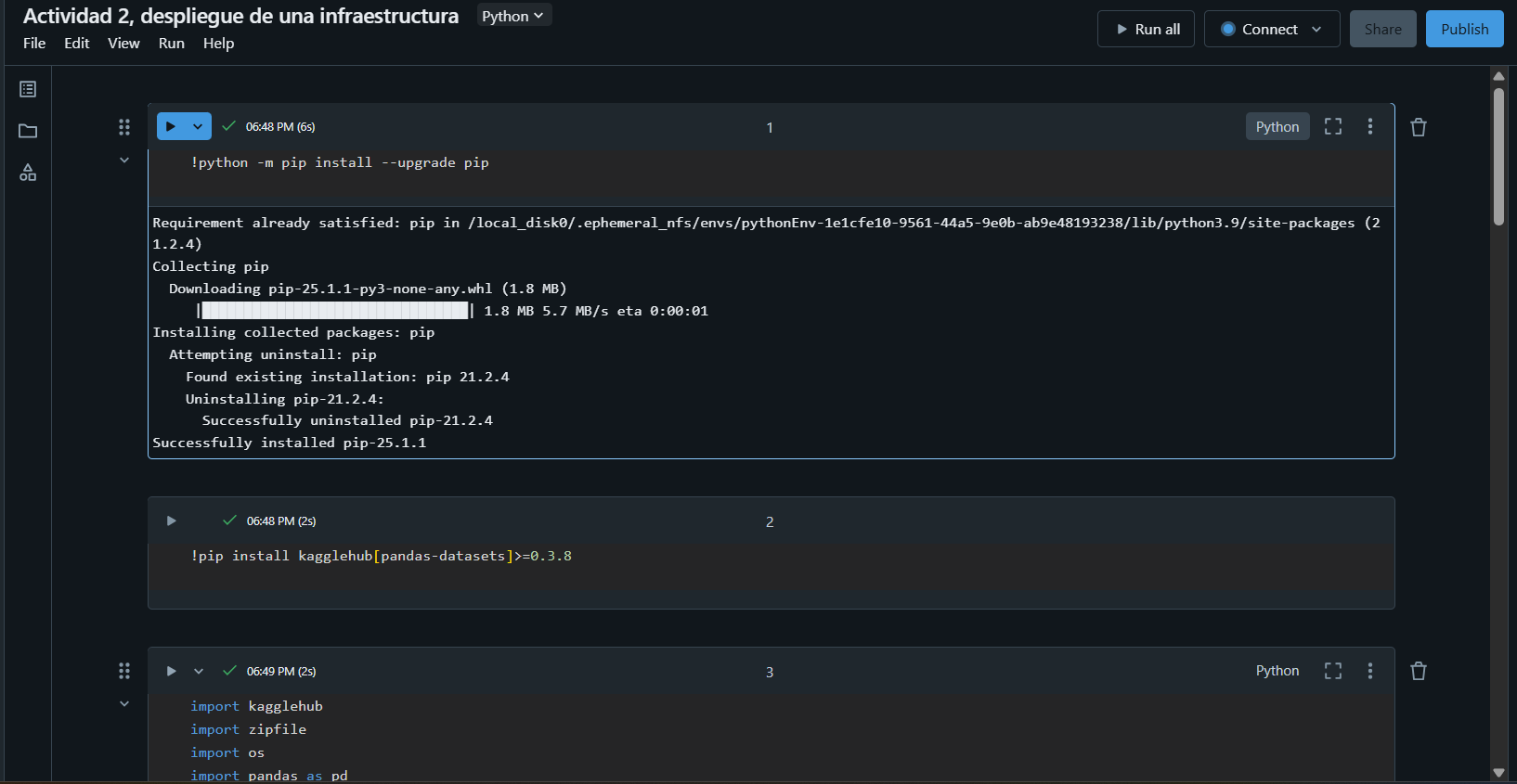
### Figura 3

### Crear un notebook Python.



### Figura 4

### Instalar pip, kagglehub, pandas.



## Despliegue y migración

Figura 5

Descargar y cargar dataset de Kaggle usando kagglehub.

Descomprimir y cargar el CSV.

Procesar y limpiar los datos (eliminación de nulos, validación de tipos).

Crear esquema y tabla en Spark DataFrame.

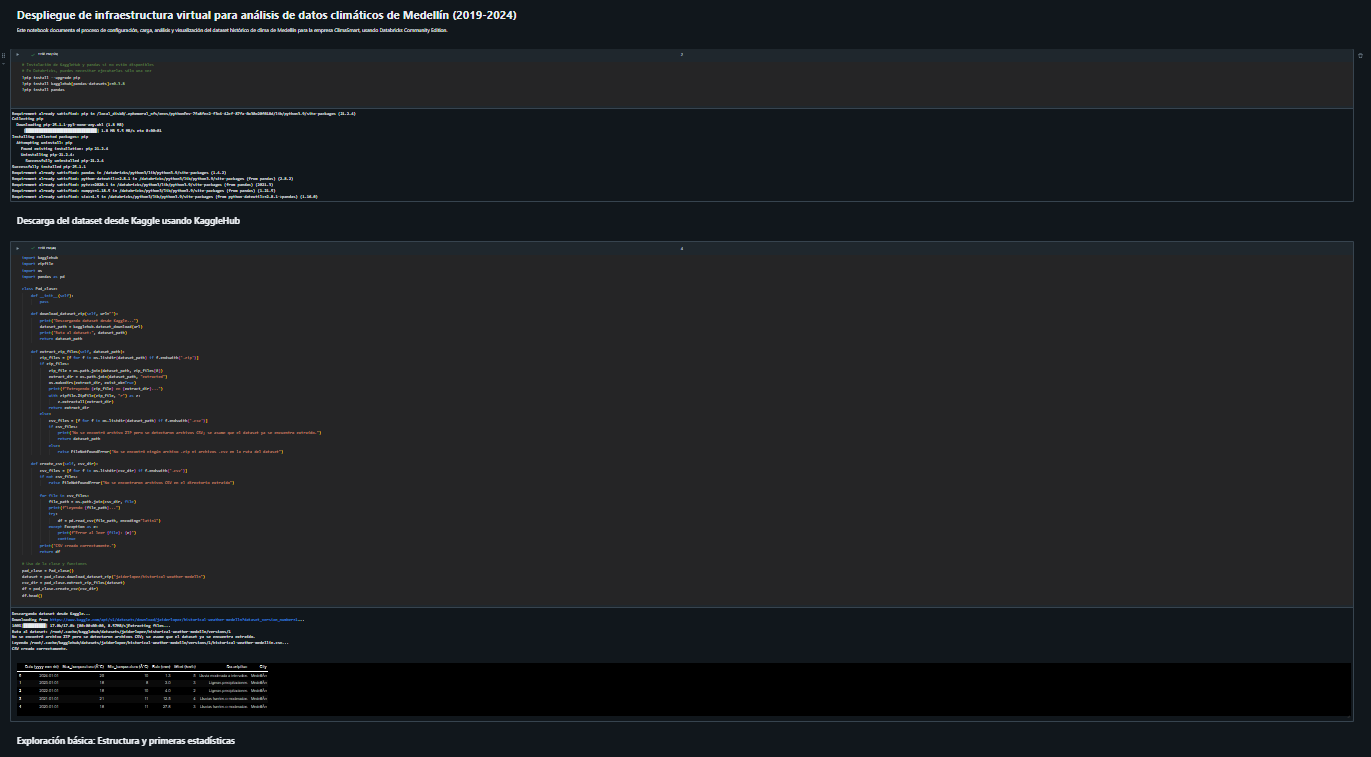
Visualizar registros y estructura de la tabla.

# Metodología empleada.

La implementación del proyecto siguió una metodología estructurada en fases claramente definidas:

**Revisión y limpieza de datos:** Identificación de preguntas relevantes, eliminación de columnas vacías o irrelevantes.

### Figura 5



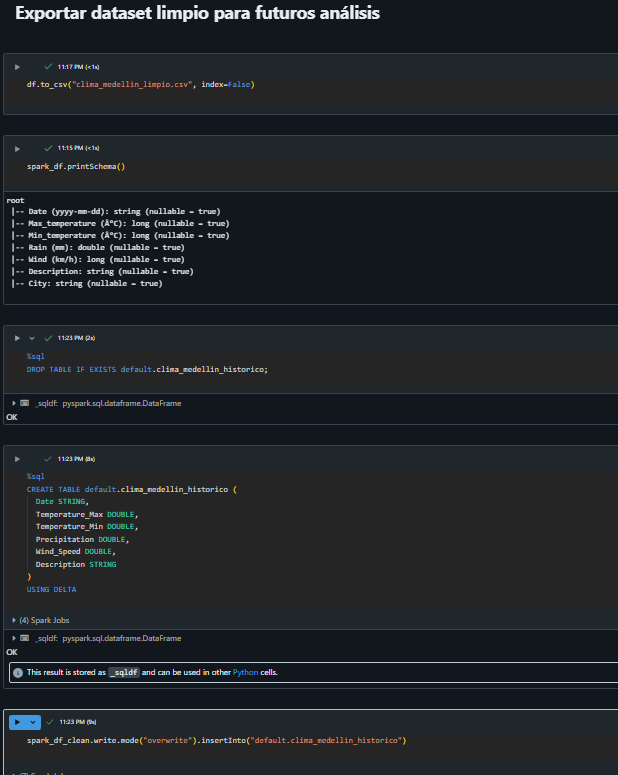
**Traducción y normalización:** Traducción técnica de los textos al español conservando el sentido original.

### Figura 6



**Implementación del sistema:** Creación de las tablas en MySQL, carga de datos con scripts en Python, generación de consultas clave.

### Figura 7



Esta metodología responde directamente a los objetivos del proyecto, priorizando la usabilidad del sistema y su adaptabilidad.

## Link repositorio gitHub:

<https://github.com/YonierAlexisQuiceno/bigData_2025_1_2.git>

## Link del código.

<https://drive.google.com/file/d/1nSNoJngvvKqQh__zGXkjxHXyDuzxBWd9/view?usp=sharing>

# Resultados

Se logró desplegar una infraestructura virtual funcional en Databricks Community Edition, integrando la ingesta del dataset histórico de clima para Medellín. El dataset fue cargado, procesado y almacenado en un DataFrame de Spark, permitiendo visualización y consultas sobre los datos. Se valida la correcta creación de la tabla y la integridad de los datos importados.

Se documentaron los pasos mediante capturas de pantalla en cada fase (ver anexos). Los scripts utilizados permiten replicar el proceso en cualquier entorno Databricks Community Edition.

# Conclusiones.

El uso de Databricks Community Edition permite a empresas como ClimaSmart montar soluciones de análisis de datos sin requerir infraestructura propia, aprovechando el poder de Apache Spark y la facilidad de integración con notebooks y librerías modernas.

La migración de datos desde Kaggle fue eficiente mediante el uso de librerías específicas como kagglehub.

La arquitectura propuesta es escalable y permite futuras ampliaciones, como integración de más fuentes o análisis avanzados.

# Bibliografía (Normas APA).

Callejas Jaramillo, A. F. (2025). Documentación académica para el desarrollo de sistemas de análisis de datos con Python y SQLite. Institución Universitaria Digital de Antioquia. Asignatura: Big Data (PREICA2501B020109).

Databricks. (s.f.). Community Cloud Databricks. Recuperado de <https://community.cloud.databricks.com/>

Kaggle. (s.f.). Historical Weather Medellín Dataset. Recuperado de <https://www.kaggle.com/datasets/jaiderlopez/historical-weather-medelln>

GitHub. (s.f.). Recuperado de <https://github.com>

# Anexos

## Anexo 1

## Código fuente

# Paso 1: Actualizar pip y preparar entorno

!python -m pip install --upgrade pip

!pip install kagglehub[pandas-datasets]>=0.3.8

import kagglehub

import zipfile

import os

import pandas as pd

class Pad\_clase:

def \_\_init\_\_(self):

pass

def download\_dataset\_zip(self, url=""):

print("Descargando dataset desde Kaggle...")

dataset\_path = kagglehub.dataset\_download(url)

print("Ruta al dataset:", dataset\_path)

return dataset\_path

def extract\_zip\_files(self, dataset\_path):

zip\_files = [f for f in os.listdir(dataset\_path) if f.endswith('.zip')]

if zip\_files:

zip\_file = os.path.join(dataset\_path, zip\_files[0])

extract\_dir = os.path.join(dataset\_path, "extracted")

os.makedirs(extract\_dir, exist\_ok=True)

print(f"Extrayendo {zip\_file} en {extract\_dir}...")

with zipfile.ZipFile(zip\_file, "r") as z:

z.extractall(extract\_dir)

return extract\_dir

else:

csv\_files = [f for f in os.listdir(dataset\_path) if f.endswith('.csv')]

if csv\_files:

print("No se encontró archivo ZIP pero se detectaron archivos CSV; se asume que el dataset ya se encuentra extraído.")

return dataset\_path

else:

raise FileNotFoundError("No se encontró ningún archivo .zip ni archivos .csv en la ruta del dataset")

def create\_csv(self, csv\_dir):

csv\_files = [f for f in os.listdir(csv\_dir) if f.endswith('.csv')]

if not csv\_files:

raise FileNotFoundError("No se encontraron archivos CSV en el directorio extraído")

for file in csv\_files:

file\_path = os.path.join(csv\_dir, file)

print(f"Leyendo {file\_path}...")

try:

df = pd.read\_csv(file\_path, encoding="latin1")

except Exception as e:

print(f"Error al leer {file}: {e}")

continue

print("CSV creado correctamente.")

return df

def limpieza\_nan\_null(self, df\_datos=pd.DataFrame(), name\_col="", reemplazar=""):

if df\_datos[name\_col].isnull().sum() > 0:

df = df\_datos.copy()

df[name\_col] = df[name\_col].fillna(reemplazar)

print("Nulos reemplazados en columna:", name\_col)

return df

print("No hay nulos en", name\_col)

return df\_datos

# Ejecución de pasos:

pad\_clase = Pad\_clase()

dataset = pad\_clase.download\_dataset\_zip("jaiderlopez/historical-weather-medelln") # Corregir por el dataset correcto de Kaggle

csv\_dir = pad\_clase.extract\_zip\_files(dataset)

df = pad\_clase.create\_csv(csv\_dir)

print(df.shape, df.head())

# Limpieza ejemplo

df = pad\_clase.limpieza\_nan\_null(df, name\_col="Precipitation", reemplazar=0)