Título: Procesamiento de Datos Meteorológicos con Apache Pig, Impala y Hive

Objetivos:

- **Comprender el proceso ETL para datos meteorológicos**
- **Aplicar las capacidades de Apache Pig, Impala y Hive para el procesamiento de datos meteorológicos**
- **Evaluar el rendimiento de Apache Pig, Impala y Hive**
- **Aplicar medidas de seguridad y privacidad a los datos meteorológicos**

1ª Parte - Cloudera o Colab (se recomienda el uso de Colab)

1. Lectura y Análisis de Datos Meteorológicos:

```
-- Cargar datos desde un archivo CSV
weather_data = LOAD '/path/to/weather.csv' USING PigStorage(',') AS (
    date:chararray,
    temperature:int,
    pressure:int,
    humidity:int
);

-- Visualización básica del contenido
DUMP weather_data;
```

2. Manejo de Valores Erróneos o Faltantes en los Datos:

```
-- Filtrar registros con datos faltantes o erróneos
clean_weather_data = FILTER weather_data BY temperature IS NOT NULL AND pressure
IS NOT NULL AND humidity IS NOT NULL;

-- Opcionalmente, filtrar valores que no son realistas
valid_weather_data = FILTER clean_weather_data BY temperature > -50 AND
temperature < 50; -- Ejemplo de rango de temperatura válido
```

3. Cálculo de la Temperatura Media con Manejo de Datos Faltantes:

```
-- Calcular la temperatura media excluyendo registros incompletos
```

```
grouped_data = GROUP valid_weather_data ALL;
average_temperature = FOREACH grouped_data GENERATE
AVG(valid_weather_data.temperature) AS avg_temp;

DUMP average_temperature;
```

4. Transformaciones Avanzadas y Unión de Datos:

```
-- Cargar otra fuente de datos, por ejemplo, datos de ubicación
location_data = LOAD '/path/to/location.csv' USING PigStorage(',') AS (
    date:chararray,
    location:chararray
);

-- Unir datos meteorológicos con datos de ubicación
joined_data = JOIN valid_weather_data BY date, location_data BY date;

-- Aplicar transformaciones o filtros adicionales según sea necesario
-- Por ejemplo, seleccionar sólo los datos de una ubicación específica
filtered_data = FILTER joined_data BY location == 'Las Palmas';

DUMP filtered_data;
```

5. Exportación de Datos para Visualización:

Para exportar los datos para su visualización en herramientas externas, como Tableau o Python, primero debes guardar los resultados en un archivo:

```
-- Guardar los resultados en un archivo
STORE filtered_data INTO '/path/to/output' USING PigStorage(',');
```

Pig almacena los datos en uno o varios archivos de texto y no en un único archivo. Esto se debe a que Pig está diseñado para trabajar con sistemas de archivos distribuidos como HDFS, donde las operaciones de almacenamiento pueden generar múltiples archivos de salida, especialmente cuando se trabaja con grandes volúmenes de datos. Para ello será necesario ejecutar los siguientes comandos para obtener el archivo de salida:

```
-- movemos el archivo de salida y lo cambiamos de nombre
!mv /content/output2/part-r-00000 /content/output.csv
```

Utilizar una herramienta de visualización para importar y visualizar estos datos. Uso de Python con Matplotlib:

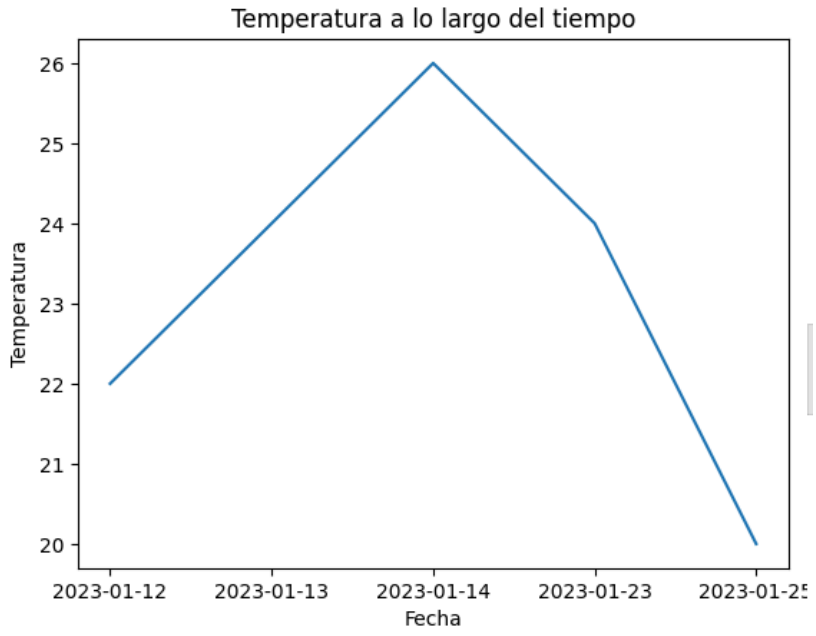
```
import pandas as pd
import matplotlib.pyplot as plt

# Cargar los datos, seleccionamos las dos primeras columnas
data = pd.read_csv('/path/to/output.csv', header=None, usecols=[0, 1])
```

Posteriormente

```
# Asignamos nombres a estas dos columnas
data.columns = ['date', 'temperature']
# Visualizar los datos temporales de temperatura
plt.plot(data['date'], data['temperature'])
plt.title('Temperatura a lo largo del tiempo')
plt.xlabel('Fecha')
plt.ylabel('Temperatura')
plt.show()
```

Deberíamos obtener una salida como esta:



Parte 2 - Ejecutar en Cloudera

1. Creación y Manipulación de Tablas de Datos Meteorológicos (uso de Hive e Impala):

Creación de la Tabla:

```
-- Crear una tabla en Hive Metastore
CREATE TABLE weather_data (
    date STRING,
    temperature INT,
    pressure INT,
    humidity INT
)
STORED AS PARQUET;
```

Inserciones Básicas:

```
-- Insertar algunos registros en la tabla
INSERT INTO weather_data (date, temperature, pressure, humidity)
VALUES
('2024-01-01', 25, 1013, 80),
('2024-01-02', 22, 1012, 82),
('2024-01-03', 24, 1014, 78),
('2024-01-04', 26, 1015, 75),
('2024-01-05', 28, 1017, 71),
('2024-01-06', 27, 1016, 74),
('2024-01-07', 26, 1015, 76),
('2024-01-08', 25, 1014, 79),
('2024-01-09', 22, 1013, 81),
('2024-01-10', 24, 1015, 80),
('2024-01-11', 26, 1016, 78),
('2024-01-12', 25, 1015, 77),
('2024-01-13', 27, 1017, 73),
('2024-01-14', 29, 1019, 69),
('2024-01-15', 30, 1020, 65),
('2024-01-16', 28, 1018, 70),
('2024-01-17', 27, 1017, 72),
('2024-01-18', 25, 1015, 75),
('2024-01-19', 23, 1014, 79),
('2024-01-20', 24, 1015, 77),
('2024-01-21', 26, 1017, 74),
('2024-01-22', 28, 1019, 71),
('2024-01-23', 27, 1018, 73),
('2024-01-24', 25, 1016, 76),
('2024-01-25', 23, 1015, 78);
```

Discusión sobre la Estructura y el Diseño:

- La tabla `weather_data` contiene columnas para la fecha, temperatura, presión y humedad.
- Se ha elegido el formato PARQUET para un almacenamiento eficiente y un mejor rendimiento en las consultas.
- Las columnas están diseñadas para capturar los aspectos esenciales de los datos meteorológicos.

2. Optimización de Consultas con Particiones y Bucketing:

Creación de Particiones (Opcional):

```
-- Crear una tabla particionada por, por ejemplo, mes
CREATE TABLE weather_data_partitioned (
    date STRING,
    temperature INT,
    pressure INT,
    humidity INT
)
PARTITIONED BY (month STRING)
STORED AS PARQUET;
```

Bucketing (Opcional):

```
-- Crear una tabla con bucketing basado en temperatura
CREATE TABLE weather_data_bucketed (
    date STRING,
    temperature INT,
    pressure INT,
    humidity INT
)
CLUSTERED BY (temperature) INTO 10 BUCKETS
STORED AS PARQUET;
```

Demostración del Impacto en el Rendimiento:

- Realizar consultas en las tablas particionadas y bucketed, y medir el tiempo de respuesta.
- Ejemplo de consulta: `SELECT * FROM weather_data WHERE temperature > 20;`
- Comparar con los tiempos de respuesta de la tabla sin particionar ni bucketing.

3. Comparación de Rendimiento con Apache Hive:

Consultas en Apache Impala:

```
-- Ejecutar una consulta en Impala
SELECT AVG(temperature) FROM weather_data WHERE humidity > 75;
```

- Para realizar dicha consulta será necesario, si utilizamos Cloudera, volver a crear la tabla e introducir los datos. Es probable que el campo date de error, por ser palabra reservada, por lo que hay que darle otro nombre.

Consultas en Hive

```
Hive:
-- Ejecutar la misma consulta en Hive
SELECT AVG(temperature) FROM weather_data WHERE humidity > 75;
```

Discusión sobre las Diferencias de Rendimiento:

- Analizar y comparar el tiempo de ejecución y la eficiencia de las consultas en Hive y en Impala.

4. Carga de Datos y Análisis Avanzado:

Carga de Datos desde un Archivo CSV (continuamos con Hive):

```
-- Cargar datos en la tabla desde un archivo CSV
LOAD DATA INPATH '/path/to/weather.csv' INTO TABLE weather_data_partitioned
PARTITION (month='2023-01');
```

Análisis Avanzado con Subconsultas y Joins:

```
Crea una nueva tabla en tu base de datos para almacenar los datos de ubicación
desde el archivo CSV:

CREATE TABLE location_data (
    date STRING,
    location STRING
)
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ',' -- Asegúrate de usar el delimitador correcto si es
diferente
STORED AS TEXTFILE;
```

Cargar los datos desde el archivo CSV en la nueva tabla "location_data" utilizando la sentencia LOAD DATA:

```
LOAD DATA INPATH '/path/to/location.csv' INTO TABLE location_data;
```

5. Seguridad y Privacidad de los Datos en Hive, Pig e Impala:

- Investiga cada uno de estos aspectos: Permisos y Control de Acceso, Encriptación de Datos, Integración con Kerberos en el contexto de Hive, Pig e Impala. Utiliza fuentes confiables y documentación oficial de estas herramientas si es posible (citar fuentes).
- Escribe un pequeño informe en el que describas cómo Hive, Pig e Impala abordan cada uno de estos aspectos de seguridad, incluyendo ejemplos y casos de uso relevantes.
- Considera la importancia de la seguridad de los datos en entornos empresariales y cómo estas medidas de seguridad pueden ayudar a proteger la información sensible.