

Trabajo 1: Despliegue de un Lakehouse que integre datos en S3 (datalake) y el data warehouse RedShift con procesamiento en Hadoop/Spark de AWS EMR, Glue, Athena y Redshift (Spectrum y ML)

Sebastian Carvalho Salazar, ✉ scarvalhos@eafit.edu.co
Juan Felipe Cardona Arango, ✉ jfcardonaa@eafit.edu.co
Juan Sebastian Sanin Villarreal, ✉ jssaninv@eafit.edu.co
Maria Alejandra Reyes Afanador, ✉ mareyesa@eafit.edu.co

Almacenamiento y Recuperación de Información

Asesor: Edwin Nelson Montoya Munera



Universidad EAFIT
Maestría en Ciencia de Datos y Analítica
Medellín
2023

TABLA DE CONTENIDOS

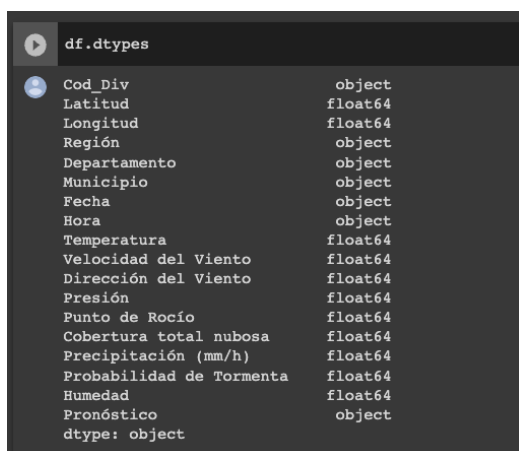
I. A). Fuentes de datos: Identificación y descarga relacionados con Cambio Climático o Calentamiento Global y datasets relacionados.	Fuentes de Datos	1
II. B)-. Ingesta de los datos como archivos / datasets desde URLs o APIs		2
II-A.	Configuración de la función lambda	2
II-B.	Ingesta de datos	3
III. C)-Diseñar un Data Lake como Almacenamiento de los datos de diferentes orígenes, tipos y estructuras.		5
III-A.	Definicion de Estructura	5
III-B.	Definicion de zonas del Data Lake	6
III-C.	Definicion de la estructura de directorio optimos	6
IV. D- Realizar catalogación en ‘raw’ y otro en ‘trusted’ de datos utilizando AWS Glue		7
IV-A.	Catalogación	7
IV-B.	Proceso ETL con AWS Glue para Ingesta en la Zona Trusted de Datos Climáticos	8
V. E. Realizar uno o dos procesos ETL para preparar los datos hacia la zona ‘trusted’ utilizando Glue y pyspark		9
VI. F) Realizar consultas básicas SQL mediante AWS Athena y Hive de los datos almacenados en el datalake mediante la catalogación realizada por AWS Glue y EMR.		10
VI-A.	Consultas SQL	10
VII. G) Realizar el modelado (deseable multidimensional) de datos / tablas para ser almacenadas en RedShift o ser accedidas desde S3 a través de RedShift Spectrum.		13
VII-A.	Configuración de Redshift Spectrum	13
VIII. H-Realizar el diseño e implementación de un ecosistema Hadoop/Spark basado en AWS EMR que permita:		14
VIII-A.	Consultas SQL en Hive y Jupyter	14
IX. Conclusiones		18
REFERENCIAS		19

LISTA DE FIGURAS

1.	Descripción de los datos	1
2.	Datos Abiertos IDEAM	1
3.	Función lambda: Automatización - Ejecución del proceso	3
4.	Función lambda: Código de ingesta de datos	4
5.	Arquitectura	5
6.	Zonas del datalake	6
7.	Glue - Schema: Base de Datos	7
8.	AWS Glue: ETL	9
9.	S3: Zona trusted	10
10.	Consulta 1: Segmentación por hora en mañana, tarde y noche	11
11.	Consulta 2: Segmentación por pronóstico	11
12.	Consulta 3: Conteo de días lluviosos por departamentos	12
13.	Configuración: Redshift Spectrum	13
14.	Configuración: Redshift Spectrum	14
15.	Consulta 3: Segmentación por hora en mañana, tarde y noche en hive	15
16.	Consulta 2: Segmentación por pronóstico en hive	15
17.	Consulta 3: Conteo de días lluviosos por departamentos en hive	16
18.	Datos: Se obtuvo la data para su análisis del s3	16
19.	consultas: Consultas de latitud por municipio y temperatura por fecha	17
20.	consultas: Consultas de municipio por fecha por temperatura y municipio por longitud y latitud	17
21.	consultas: Última consulta de municipios que registran una temperatura mayor a 30 grados	18

I. A). FUENTES DE DATOS: IDENTIFICACIÓN Y DESCARGA RELACIONADOS CON CAMBIO CLIMÁTICO O CALENTAMIENTO GLOBAL Y DATASETS RELACIONADOS.FUENTES DE DATOS

El origen de datos comprendió la extracción de fuente de datos abierto del IDEAM, el cual proporciona datos meteorológicos, información climática, pronósticos del tiempo.



```
df.dtypes
```

Cod_Div	object
Latitud	float64
Longitud	float64
Región	object
Departamento	object
Municipio	object
Fecha	object
Hora	object
Temperatura	float64
Velocidad del Viento	float64
Dirección del Viento	float64
Presión	float64
Punto de Rocío	float64
Cobertura total nubosa	float64
Precipitación (mm/h)	float64
Probabilidad de Tormenta	float64
Humedad	float64
Pronóstico	object
dtype:	object

FIG. 1: Descripción de los datos

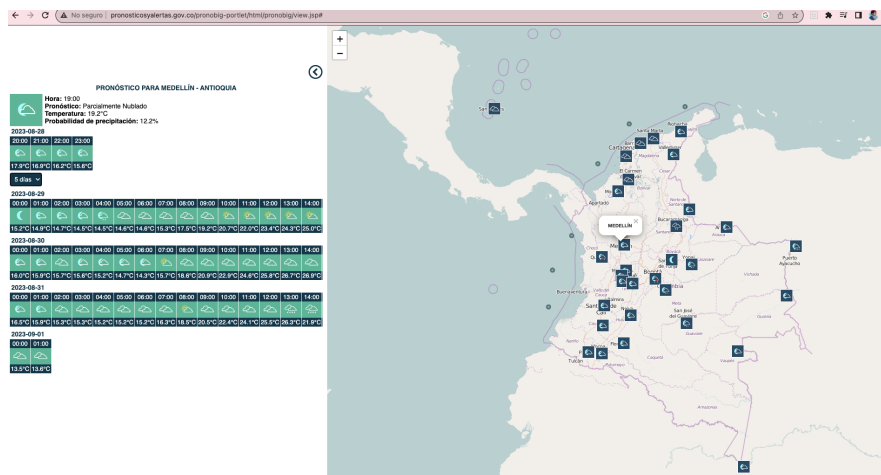


FIG. 2: Datos Abiertos IDEAM

II. B)-. INGESTA DE LOS DATOS COMO ARCHIVOS / DATASETS DESDE URLS O APIS

Para la ingesta de datos el cual comprende importar los datos de las fuentes al sistemas de almacenamiento en la nube se implementara un un robot, que descargue periodicamente los archivos y los almacene en AWS S3 (zona raw), a continuacion se describe el proceso y los pasos implementados

II-A. *Configuración de la función lambda*

AWS Lambda es un servicio de computación que ejecuta el código en respuesta a eventos y administra automáticamente los recursos de computación, en este caso se implementara una pipeline que descargue periodicamente los archivos y los almacene en AWS S3 (zona raw).

- **Modificación del Tiempo de Timeout:** Para permitir que el proceso se complete correctamente, fue necesario ajustar el tiempo por defecto del timeout de la función Lambda. Dado que las operaciones de descarga, extracción y subida a veces pueden tomar más tiempo, se aumentó el timeout para asegurar la finalización exitosa de cada etapa.
- **Añadido de Cron con EventBridge (CloudWatch Events):** Se implementó un cron utilizando EventBridge (anteriormente conocido como CloudWatch Events) para desencadenar la función Lambda de manera programada. El cron utilizado es **"28 16 ? * SUN *"**, lo que significa que la función Lambda se ejecutará a las 16:28 los domingos. Esto permite automatizar el proceso de manera regular.
- **Inclusión de Librerías en una Capa (Layer):** Para optimizar el manejo de dependencias, se decidió descargar las librerías **requests** y **urllib3**. Estas librerías son esenciales para el proceso de descarga y son utilizadas en la función Lambda. Sin embargo, para evitar la repetición y el aumento del tamaño del archivo ZIP de la función, las librerías se descargaron y subieron como un archivo ZIP a una capa (layer) de la función Lambda. Esto permite que las librerías se mantengan en una capa separada y sean compartidas entre múltiples funciones Lambda, reduciendo el tamaño del archivo principal de la función y facilitando las actualizaciones y cambios.
- **Limpieza de Caracteres Especiales y Verificación de Encoding en la Automatización de Extracción de Datos:** Dentro del proceso automatizado de extracción de datos, se incluye una etapa de limpieza de caracteres especiales como acentos y la conversión de "ñ.^a" a "n". antes de la ingesta a la zona raw, asegurando datos consistentes.

También se verifica el encoding para prevenir problemas de interpretación. Estas medidas mantienen la integridad de los datos en todo el flujo de trabajo automatizado.

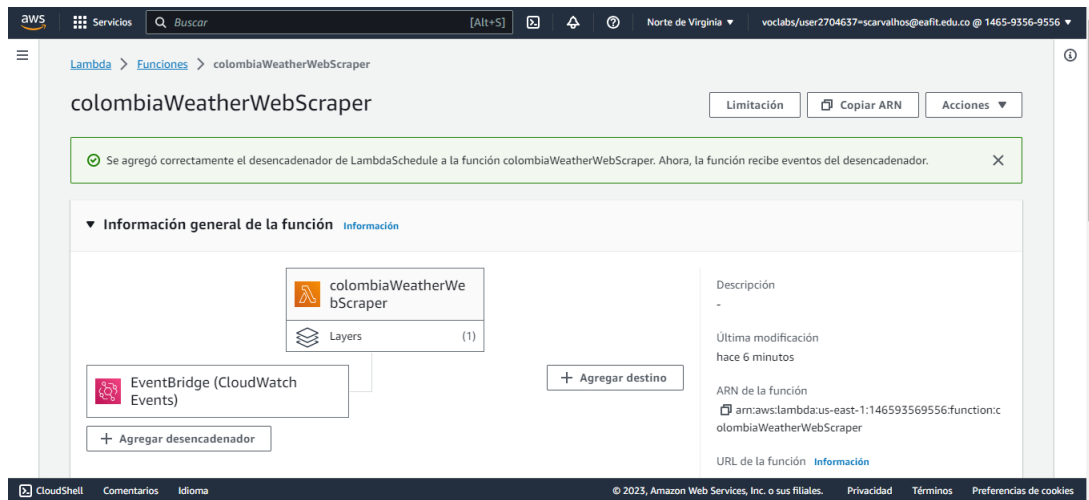


FIG. 3: Función lambda: Automatización - Ejecución del proceso

II-B. Ingesta de datos

Este informe detalla el proceso de una función AWS Lambda escrita en Python. La función se encarga de descargar un archivo ZIP desde una URL, extraer un archivo CSV de dicho ZIP, renombrarlo y finalmente, subirlo a un bucket de Amazon S3.

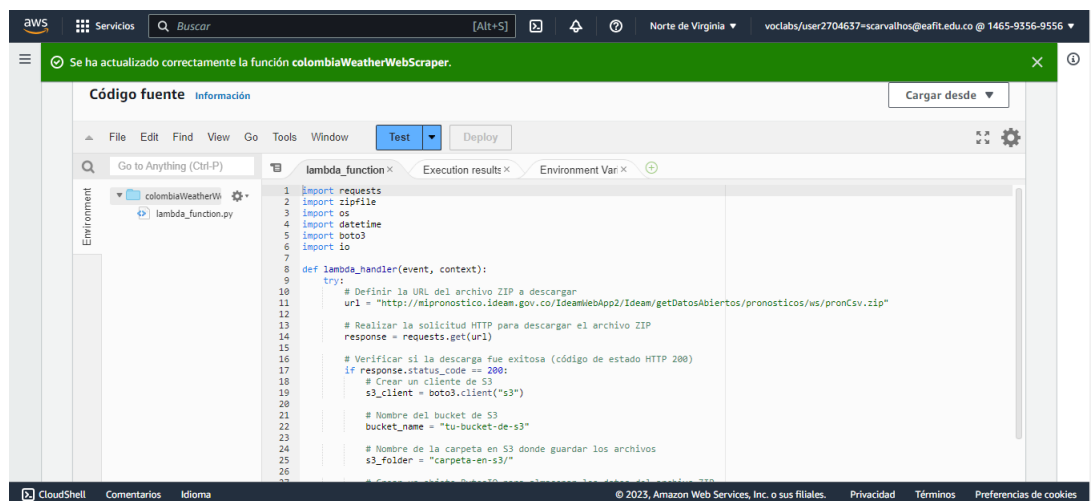
■ Importar Librerías:

- **import requests:** Permite realizar solicitudes HTTP.
- **import boto3:** Proporciona acceso a los servicios de AWS, específicamente Amazon S3.
- **import io:** Proporciona herramientas para trabajar con flujos de bytes en memoria.
- **import zipfile:** Permite trabajar con archivos ZIP.
- **import os:** Proporciona funciones para interactuar con el sistema de archivos.
- **import datetime:** Permite trabajar con fechas y horas.
- **import tempfile:** Proporciona funciones para trabajar con directorios temporales.

- **Función `lambda_handler(event, context)`:** Esta es la función principal que se ejecuta cuando el evento Lambda es desencadenado.

- **Descarga del Archivo ZIP:** Se define la URL del archivo ZIP a descargar y se utiliza la librería **requests** para realizar la solicitud HTTP.

- **Verificación de la Descarga:** Se verifica el código de estado HTTP en la respuesta para determinar si la descarga fue exitosa. Si no es exitosa, se muestra un mensaje de error.
- **Creación de un Directorio Temporal:** Se crea un directorio temporal utilizando `tempfile.TemporaryDirectory()` para extraer el contenido del ZIP.
- **Cliente de Amazon S3:** Se crea un cliente de Amazon S3 utilizando `boto3.client`.
- **Nombres de Bucket y Carpeta:** Se definen el nombre del bucket de Amazon S3 y el nombre de la carpeta donde se guardarán los archivos.
- **Almacenamiento en Memoria del Archivo ZIP:** Se crea un objeto `io.BytesIO` para almacenar los datos del archivo ZIP descargado.
- **Extracción del Archivo ZIP:** Se utiliza la librería `zipfile` para extraer el contenido del archivo ZIP. Se filtran los nombres de archivo dentro del ZIP para obtener los que tengan extensión `.csv`.
- **Renombrar y Subir el Archivo CSV:** Se renombra el archivo CSV extraído con la fecha actual y se añade el nuevo nombre al archivo. Luego, el archivo CSV renombrado se sube al bucket de S3 utilizando `s3_client.upload_file`.
- **Manejo de Excepciones:** Si ocurre alguna excepción durante el proceso, se captura y se retorna un mensaje de error.



```
1 import requests
2 import zipfile
3 import os
4 import datetime
5 import boto3
6 import io
7
8 def lambda_handler(event, context):
9     try:
10         # Definir la URL del archivo ZIP a descargar
11         url = "http://mipronostico.ideam.gov.co/IdeamWebApp2/Ideam/getDatosAbiertos/pronosticos/ws/pronCsv.zip"
12
13         # Realizar la solicitud HTTP para descargar el archivo ZIP
14         response = requests.get(url)
15
16         # Verificar si la descarga fue exitosa (código de estado HTTP 200)
17         if response.status_code == 200:
18             # Crear un cliente de S3
19             s3_client = boto3.client("s3")
20
21             # Nombre del bucket de S3
22             bucket_name = "tu-bucket-de-s3"
23
24             # Nombre de la carpeta en S3 donde guardar los archivos
25             s3_folder = "carpeta-en-s3/"
26
27             # Extraer el contenido del ZIP
28             with zipfile.ZipFile(io.BytesIO(response.content)) as zip_file:
29                 for file in zip_file.namelist():
30                     # Filtrar archivos con extensión .csv
31                     if file.endswith(".csv"):
32                         # Renombrar el archivo con la fecha actual
33                         filename = f"{datetime.datetime.now().strftime('%Y%m%d%H%M%S')}.csv"
34                         # Subir el archivo a S3
35                         s3_client.upload_file(filename, bucket_name, s3_folder + filename)
```

FIG. 4: Función lambda: Código de ingesta de datos

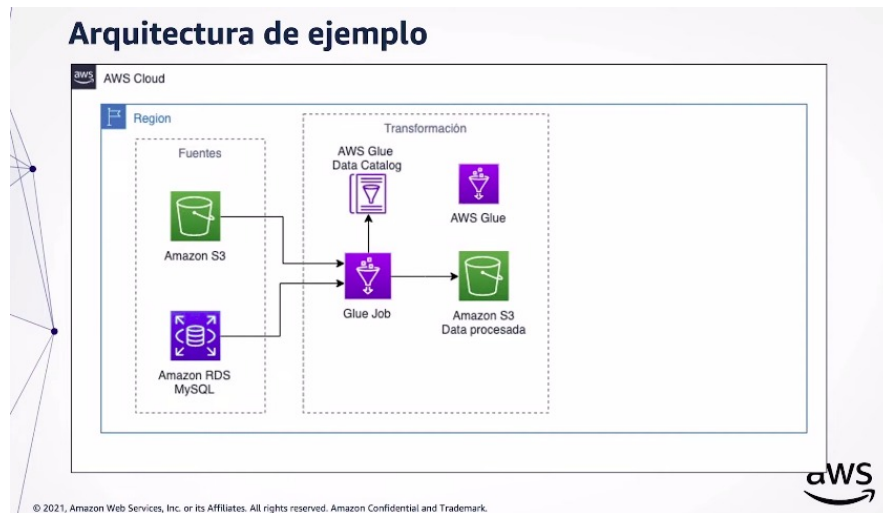


FIG. 5: Arquitectura

III. C)-DISEÑAR UN DATA LAKE COMO ALMACENAMIENTO DE LOS DATOS DE DIFERENTES ORÍGENES, TIPOS Y ESTRUCTURAS.

III-A. Definición de Estructura

Se implementará una estructura por tipo BATCH dado que los datos meteorológicos de las fuentes se generan en tiempos espaciados. (ej. cada 15 minutos, cada 3 horas, o diario).

■ Data Source

Comprende los repositorios de origen de los datos sin procesar de las diversas fuentes.

- **Ingesta:(Ingest ETL)** Aquí es donde los datos se recopilan de diversas fuentes y se ingieren en el Data Lake.
- **Almacenamiento:(Data Store)** Los datos ingresados se almacenan en su forma bruta en esta capa. La ventaja del Data Lake es que no se requiere una estructura predefinida. Los sistemas de almacenamiento escalables como Hadoop HDFS, Amazon S3.
- **Procesamiento y Acceso(Data Processing):** Aquí se lleva a cabo el procesamiento de datos y se permite el acceso para análisis y consulta. Las tecnologías como Apache Spark, Presto, Hive y herramientas de análisis en la nube pueden usarse para procesar y consultar datos.

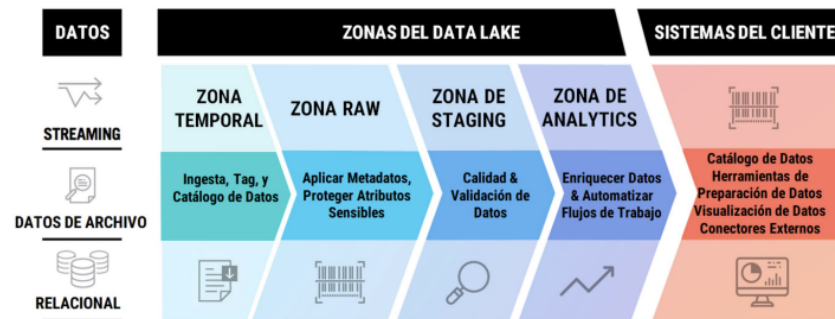


FIG. 6: Zonas del datalake

III-B. Definición de zonas del Data Lake

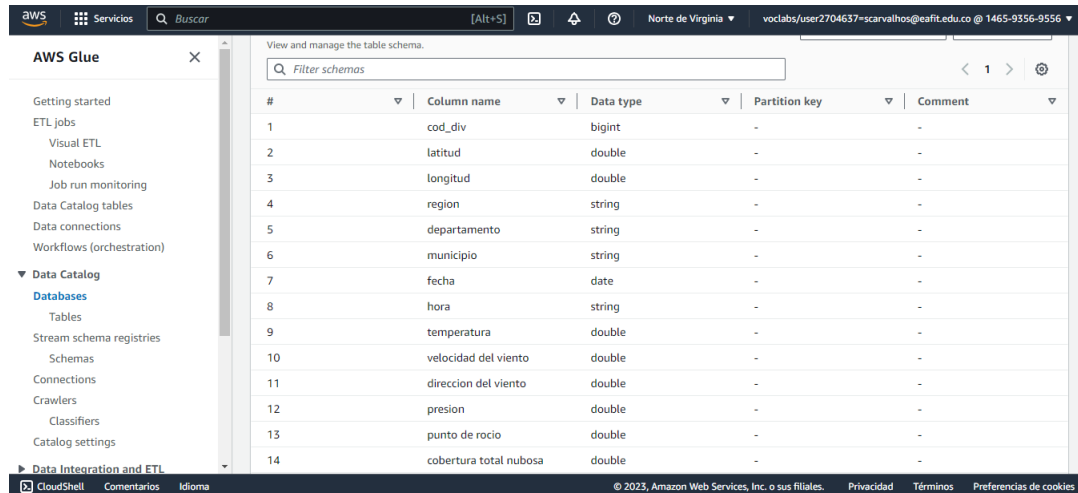
- **Raw Zone:** Aquí se almacenarán los datos en su formato original, tal como se ingresaron, sin procesamiento, siendo un directorio de referencial útil como copia sin cambios de los datos para referencias futuras.
- **Staging Zone:** En esta zona, los datos son transformados y preparados para su procesamiento y análisis posterior. Se incluyen limpiezas, normalizaciones y transformaciones.
- **Curated Zone:** Aquí se almacenan los datos limpios, con transformaciones y estructurados, listos para su análisis. y consulta se puede organizar en tablas.

III-C. Definición de la estructura de directorio optimos

Considerando la naturaleza de los datos el cual está relacionado con el tiempo, se implementa una organización por fecha: como `/ingestas/2023/08/28` ”

IV. D- REALIZAR CATALOGACIÓN EN ‘RAW’ Y OTRO EN ‘TRUSTED’ DE DATOS UTILIZANDO AWS GLUE

IV-A. Catalogación



#	Column name	Data type	Partition key	Comment
1	cod_div	bigint	-	-
2	latitud	double	-	-
3	longitud	double	-	-
4	region	string	-	-
5	departamento	string	-	-
6	municipio	string	-	-
7	fecha	date	-	-
8	hora	string	-	-
9	temperatura	double	-	-
10	velocidad del viento	double	-	-
11	direccion del viento	double	-	-
12	presion	double	-	-
13	punto de rocío	double	-	-
14	cobertura total nubosa	double	-	-

FIG. 7: Glue - Schema: Base de Datos

El proceso de catalogación en AWS Glue implica definir la estructura y propiedades de los datos almacenados en un almacén de datos, lo que facilita su descubrimiento y análisis. En este caso, se catalogan datos meteorológicos con las siguientes características:

- **Columnas:** Se enumeran las columnas en los datos.
- **Data Type:** Indica el tipo de datos que cada columna contiene (por ejemplo, bigint, double, string, date).
- **Partition Key:** No se establece ninguna clave de partición, lo que significa que no se están utilizando particiones para organizar los datos.
- **Comment:** No se proporciona ningún comentario para las columnas en este caso.

Las columnas catalogadas son características de los datos meteorológicos y están organizadas de la siguiente manera:

- Códigos de división geográfica (**cod_div**), latitud y longitud.
- Información de ubicación (región, departamento y municipio).
- Fecha, hora y pronóstico relacionado.
- Mediciones meteorológicas como temperatura, velocidad del viento, dirección del viento, presión, punto de rocío, cobertura nubosa, precipitación, probabilidad de tormenta y humedad.

Este proceso de catalogación en AWS Glue permite a los usuarios tener una vista organizada y estructurada de los datos meteorológicos almacenados, lo que facilita la consulta y el análisis posterior [1].

IV-B. Proceso ETL con AWS Glue para Ingesta en la Zona Trusted de Datos Climáticos

En el contexto de habilitar la ingesta efectiva de datos en la Zona Trusted, se ha implementado un proceso ETL (Extract, Transform, Load) utilizando AWS Glue, una herramienta de servicios gestionados para transformación de datos en la nube. Este proceso permite tomar datos brutos, realizar transformaciones esenciales y finalmente cargarlos en la Zona Trusted para análisis y uso.

Extracción (Extract):

Se inicia tomando como fuente de datos la tabla **colombia_weather_ideam**, previamente catalogada en el entorno de AWS. Esta tabla contiene información climática detallada para diferentes departamentos y fechas en Colombia. AWS Glue se conecta a esta fuente y extrae los datos necesarios para el proceso de transformación.

Transformación (Transform):

En esta etapa, se realiza la transformación de los datos para adaptarlos a las necesidades y requisitos de la Zona Trusted. El primer paso es ajustar el esquema de la tabla eliminando columnas que no son relevantes para el análisis. Esto permite reducir la complejidad y el volumen de datos a procesar.

Enriquecimiento de Datos (Transform):

El punto clave del proceso es la aplicación de una consulta específica que enriquece los datos. Esta consulta calcula estadísticas significativas a partir de los atributos climáticos, como promedios de temperatura, velocidad del viento, presión, cobertura nubosa y humedad. Además, se determina la cantidad de días lluviosos utilizando una lógica condicional. Estos cálculos proporcionan información agregada y valiosa para análisis posteriores.

Agrupación y Ordenamiento (Transform):

Los resultados de la consulta se agrupan por **Departamento** y **Fecha**, lo que facilita la comprensión y el análisis de tendencias climáticas a lo largo del tiempo en diferentes regiones. Además, se ordenan cronológicamente para permitir una visualización coherente de los datos.

Carga (Load):

Finalmente, los datos transformados y enriquecidos se cargan en la Zona Trusted. Esta zona está diseñada para albergar datos listos para análisis y toma de decisiones, brindando un entorno seguro y confiable para su acceso por parte de los usuarios autorizados.

V. E. REALIZAR UNO O DOS PROCESOS ETL PARA PREPARAR LOS DATOS HACIA LA ZONA 'TRUSTED' UTILIZANDO GLUE Y PYSPARK

El proceso ETL con AWS Glue, que abarca la extracción, transformación y carga de datos climáticos, se convierte en una parte integral de la infraestructura de datos. Al eliminar la complejidad y realizar transformaciones específicas, se optimiza la calidad y utilidad de los datos antes de su ingestión en la Zona Trusted, permitiendo que los usuarios obtengan información valiosa de manera eficiente y precisa.

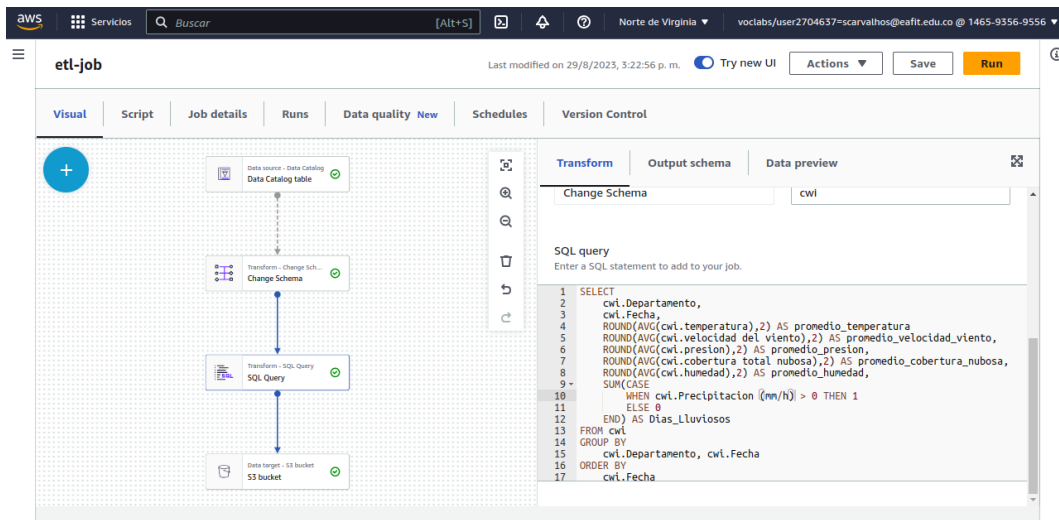


FIG. 8: AWS Glue: ETL

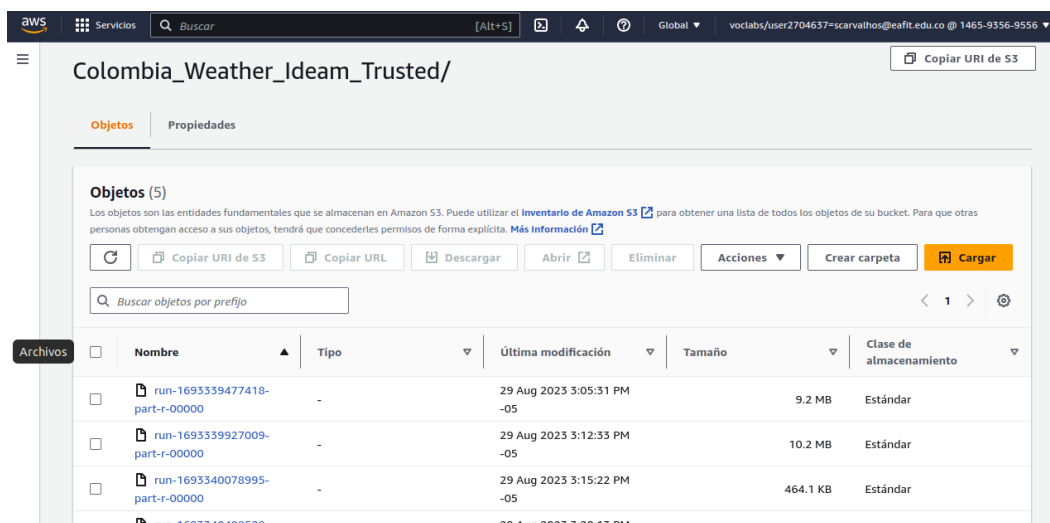


FIG. 9: S3: Zona trusted

En este escenario específico, se optó por implementar el ETL utilizando una forma de ejecución bajo demanda. Esto significa que el proceso de Extracción, Transformación y Carga se activa y ejecuta únicamente cuando se solicita. Esta elección ofrece flexibilidad al permitir que el proceso ETL se inicie manualmente según las necesidades, en lugar de ejecutarse de manera programada y automatizada en intervalos regulares. Esta estrategia de ejecución bajo demanda es beneficiosa cuando se necesita mayor control sobre el proceso y cuando las transformaciones pueden variar según los requerimientos cambiantes.

VI. F) REALIZAR CONSULTAS BÁSICAS SQL MEDIANTE AWS ATHENA Y HIVE DE LOS DATOS ALMACENADOS EN EL DATALAKE MEDIANTE LA CATALOGACIÓN REALIZADA POR AWS GLUE Y EMR.

VI-A. Consultas SQL

En esta sección, se proporciona una explicación detallada del análisis climático realizado mediante consultas SQL en la base de datos.

■ Primera consulta:

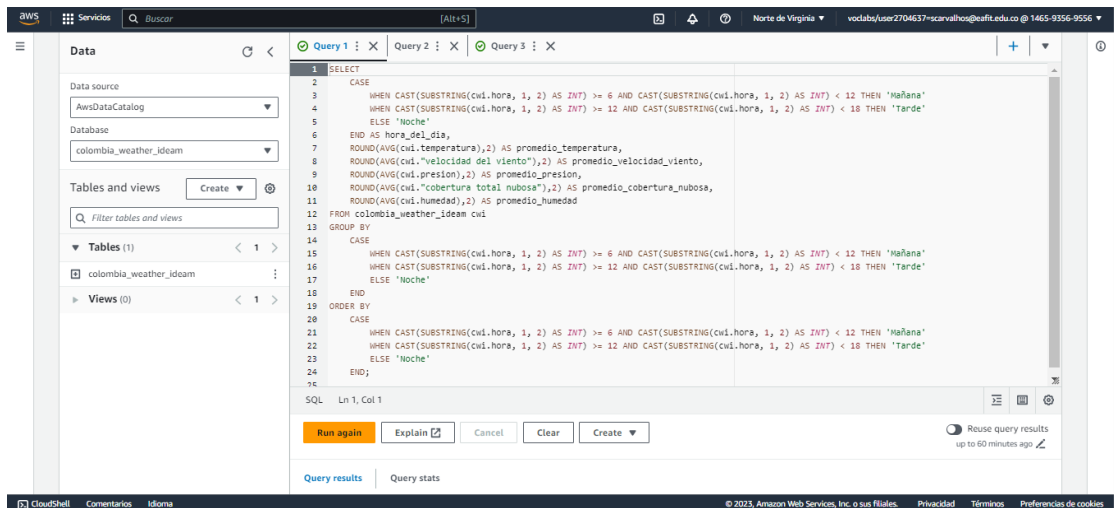


FIG. 10: Consulta 1: Segmentación por hora en mañana, tarde y noche

Esta consulta de SQL calcula los promedios de datos meteorológicos en Colombia para diferentes momentos del día (mañana, tarde y noche). Luego muestra estos promedios junto con la clasificación de cada registro en uno de esos momentos. Los resultados se agrupan y ordenan según los momentos del día.

■ Segunda consulta:

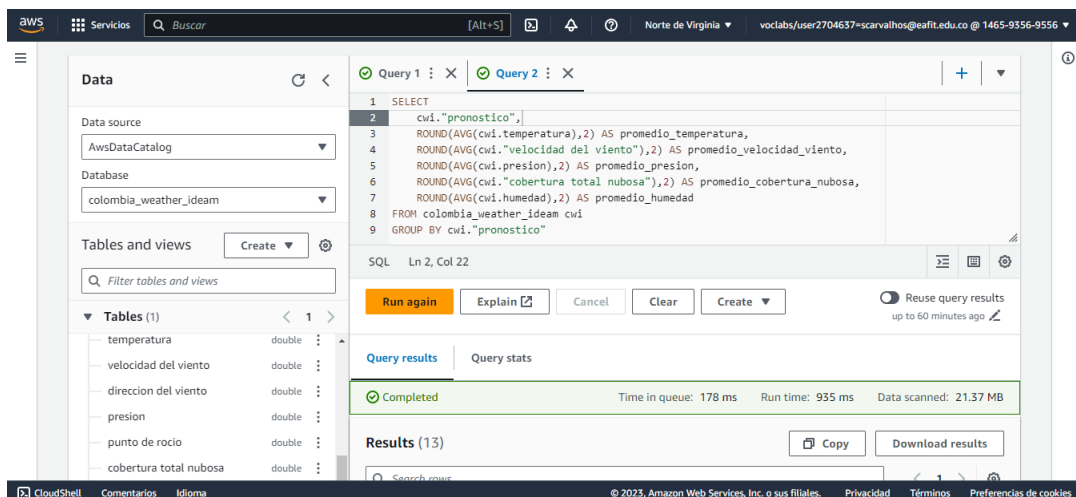


FIG. 11: Consulta 2: Segmentación por pronóstico

Esta consulta SQL toma información meteorológica de diferentes departamentos en Colombia y calcula los promedios de temperatura, velocidad del viento, presión, cobertura nubosa y humedad para cada departamento. Luego muestra estos promedios junto con el nombre de cada departamento en el resultado. La información se agrupa por departamento para realizar los cálculos de promedio de manera separada para cada área geográfica.

■ Tercera consulta:

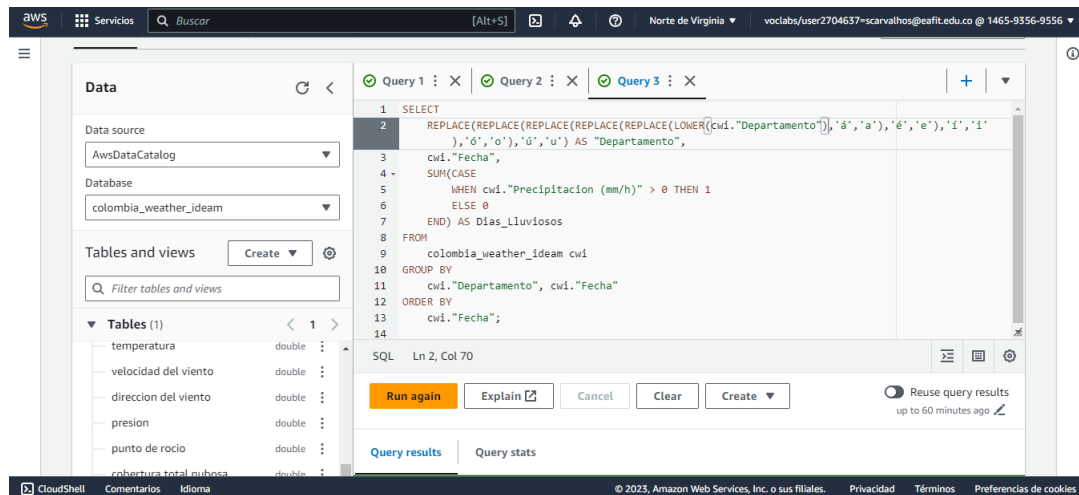


FIG. 12: Consulta 3: Conteo de días lluviosos por departamentos

La consulta se centra en determinar la cantidad de días lluviosos por departamento y fecha. Además, normaliza los nombres de los departamentos para garantizar que los resultados sean coherentes independientemente de la presencia de acentos o variaciones en mayúsculas y minúsculas.

VII. G) REALIZAR EL MODELADO (DESEABLE MULTIDIMENSIONAL) DE DATOS / TABLAS PARA SER ALMACENADAS EN REDSHIFT O SER ACCEDIDAS DESDE S3 A TRAVÉS DE REDSHIFT SPECTRUM.

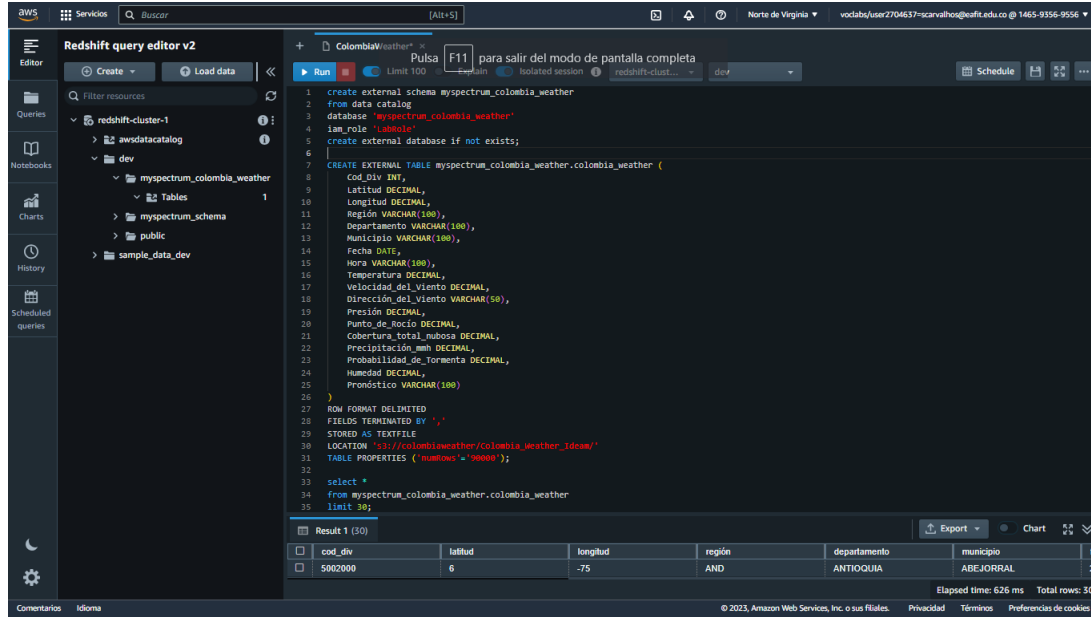


FIG. 13: Configuración: Redshift Spectrum

VII-A. Configuración de Redshift Spectrum

Para aprovechar Amazon Redshift Spectrum y realizar consultas en datos externos almacenados en Amazon S3, es necesario crear un esquema y una tabla externa. Los siguientes pasos describen el proceso de configuración:

- **Crear un Esquema Externo:** Se establece un esquema externo llamado **myspectrum_colombia_weather** para hacer referencia a los datos externos en el catálogo de datos. Este esquema está asociado con la base de datos **myspectrum_colombia_weather** y utiliza un rol de Identidad y Acceso de AWS (IAM) para permisos de acceso. Si la base de datos externa no existe, se creará.
- **Crear una Tabla Externa:** Se crea una tabla externa llamada **colombia_weather** dentro del esquema **myspectrum_colombia_weather**. Esta tabla se corresponde con la estructura de los datos externos y define los tipos de datos de columna, como enteros, decimales y cadenas de caracteres. Los datos se almacenan en un formato de texto delimitado por comas. La ubicación de los datos se especifica como **s3://colombiaweather/Colombia_Weather_Ideam/**. Además, se establecen propiedades de tabla como la cantidad de filas para optimizar el rendimiento de las consultas.

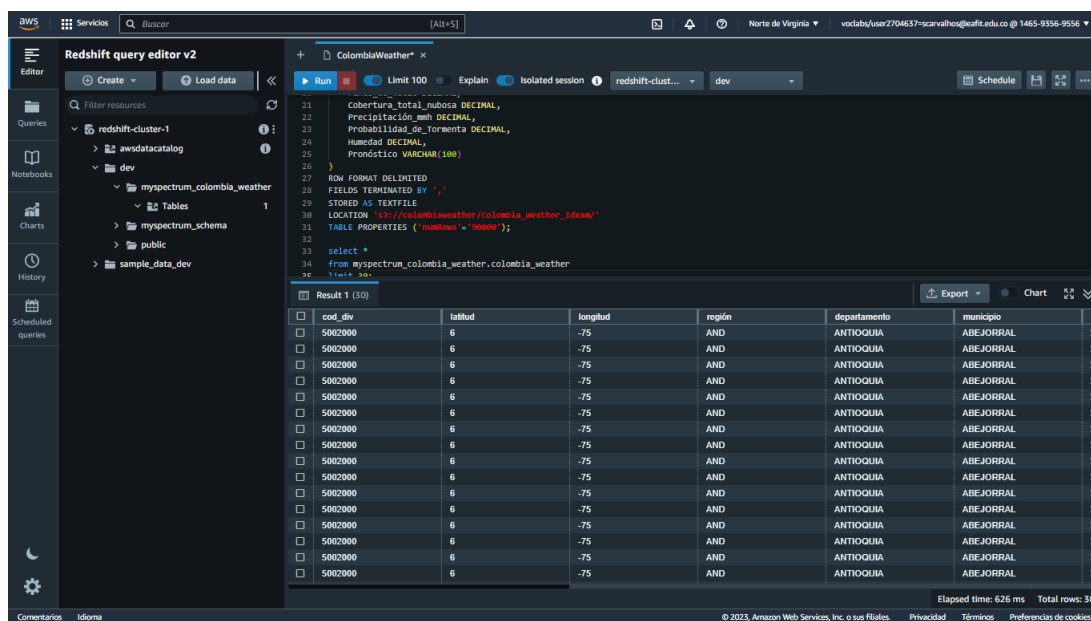


FIG. 14: Configuración: Redshift Spectrum

VIII. H-REALIZAR EL DISEÑO E IMPLEMENTACIÓN DE UN ECOSISTEMA HADOOP/SPARK BASADO EN AWS EMR QUE PERMITA:

VIII-A. Consultas SQL en Hive y Jupyter

En esta sección se representan las consultas definidas en la sección V en los notebooks de Jupyter alojados en el S3 y las consultas en Hive.

Cabe resaltar que para una mayor facilidad, se realizó cambios en los nombres de algunas columnas a snake case, los cuales son: velocidad del viento, cobertura total nuboso, dirección del viento y probabilidad de tormenta.

■ Primera consulta:

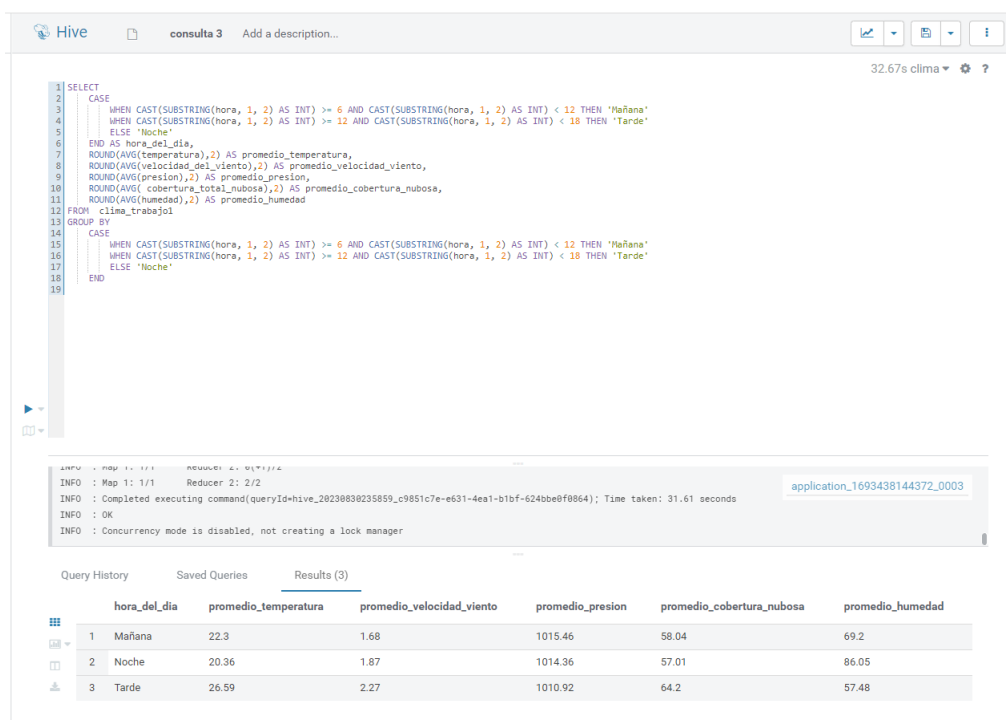


FIG. 15: Consulta 3: Segmentación por hora en mañana, tarde y noche en hive

■ Segunda consulta:

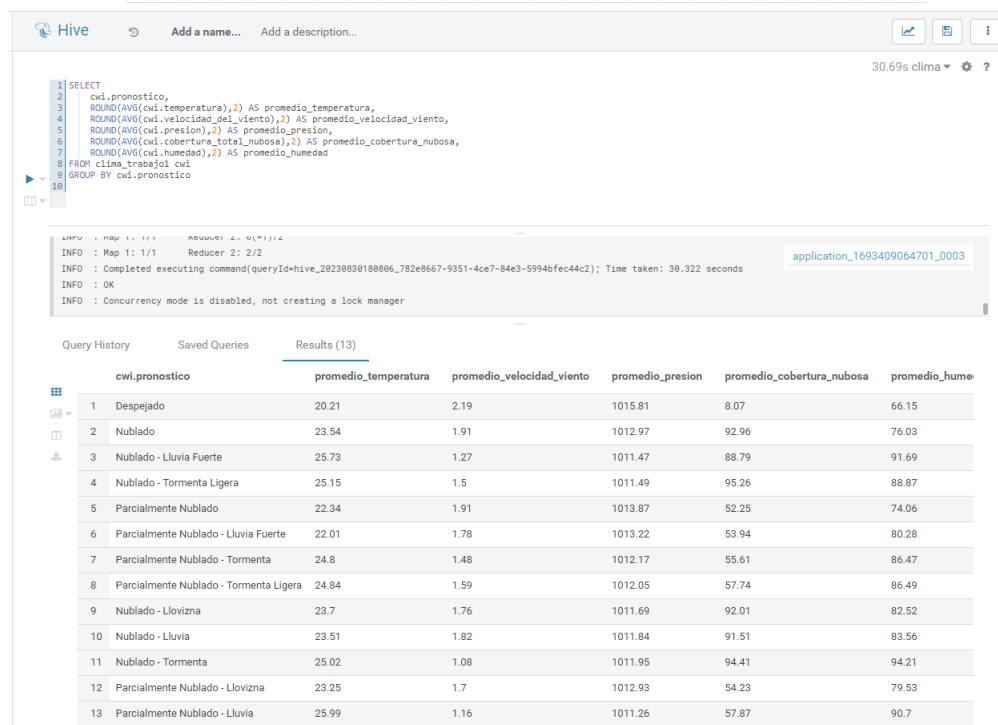


FIG. 16: Consulta 2: Segmentación por pronóstico en hive

■ Tercera consulta:

■ Primeras 2 consultas:

```
In [4]: df.createOrReplaceTempView("clima")
spark.sql("select latitud,municipio from clima limit 10").show()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

latitud	municipio
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL
5.789301	ABEJORRAL

```
In [5]: spark.sql("select fecha,temperatura from clima ORDER BY temperatura ").show()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

fecha	temperatura
2023-08-28	3.2
2023-08-28	3.3
2023-08-28	3.3
2023-08-28	3.3
2023-08-28	3.4
2023-08-28	3.4
2023-08-28	3.5
2023-08-28	3.5
2023-08-28	3.5
2023-08-28	3.5
2023-08-28	3.5

FIG. 19: consultas: Consultas de latitud por municipio y temperatura por fecha

■ Segundas 2 consultas:

```
In [6]: spark.sql("select municipio,fecha,temperatura from clima where temperatura > 20 ORDER BY temperatura ").show()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

municipio	fecha	temperatura
SABANA	2023-08-29	20.1
ARCABUCO	2023-08-29	20.1
SAN LUIS	2023-08-28	20.1
CABRERA	2023-08-30	20.1
SABANA	2023-08-28	20.1
ARGENTA	2023-08-27	20.1
ROLDANILLO	2023-08-28	20.1
ABEJORRAL	2023-08-27	20.1
SABANA	2023-08-29	20.1
ARIENTA	2023-08-28	20.1
SABANALARGA	2023-08-29	20.1
ABRIQUA	2023-08-30	20.1
SABANA	2023-08-30	20.1
ARIENTA	2023-08-29	20.1
SABANALARGA	2023-08-28	20.1
AGRADO	2023-08-30	20.1
SAN ANDRÉS	2023-08-28	20.1
BALBOA	2023-08-28	20.1
SABANALARGA	2023-08-28	20.1
AMAGA	2023-08-29	20.1

only showing top 20 rows

```
In [7]: spark.sql("select municipio,latitud,longitud from clima ORDER BY latitud ").show()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

municipio	latitud	longitud
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582
LETICIA	-4.198445	-69.941582

FIG. 20: consultas: Consultas de municipio por fecha por temperatura y municipio por longitud y latitud

■ Última consulta:

```
In [11]: spark.sql("select municipio from clima where temperatura>30 GROUP BY municipio").show()

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available,
or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

+-----+
| municipio |
+-----+
| GUTIÉRREZ |
| PAICOL |
| ANSERIMUEVO |
| LA PEDRERA |
| GARAGOA |
| PELAYA |
| PISBA |
| RÍO QUITO |
| GAMA |
| LA SALTINA |
| LÓPEZ DE VICAY |
| PURZA |
| PUERTO ARICA |
| RETIRO |
| PORE |
| CONFINAS |
| CAICEDO |
| NETRA |
| PINILLOS |
| RIOBLANCO |
+-----+
only showing top 20 rows

In [ ]:
```

FIG. 21: consultas: Última consulta de municipios que registran una temperatura mayor a 30 grados

IX. CONCLUSIONES

- Se profundizó las arquitecturas de una solución big data de un proveedor de Nube(AWS).
- Se desarrolló un caso de estudio que aplique las etapas de un ciclo de vida de un proyecto de Big Data Analytics.

REFERENCIAS

- [1] AWS. «AWS Documentation.» Find user guides, code samples, SDKs and toolkits, tutorials, API and CLI references, and more. (2023), dirección: <https://docs.aws.amazon.com/>.