

**UNIVERSIDAD EAFIT**  
**ST1800/ST1801 ALMACENAMIENTO Y RECUPERACIÓN DE INFORMACIÓN**  
**TRABAJO #1**

**2023-2**

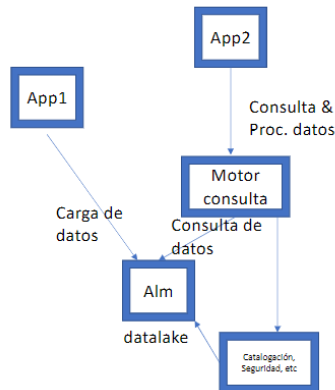
**Nombre: Pablo Simón Arias Zuluaga - email: [psariasz@eafit.edu.co](mailto:psariasz@eafit.edu.co)**

**Nombre: Sebastián Giraldo Gómez - email: [sgiraldog7@eafit.edu.co](mailto:sgiraldog7@eafit.edu.co)**

**[https://github.com/psariasz/Trabajo1\\_almacenamiento](https://github.com/psariasz/Trabajo1_almacenamiento)**

Para el presente trabajo se creará el concepto de DataLake en AWS. Donde almacenaremos los [datos hidrometeorológicos crudos – Red de Estaciones IDEAM: Temperatura](#).

El proceso que vamos a llevar a cabo con estos datos, es un proceso en batch donde se extraen los datos directamente desde su fuente en este caso la pagina de datos abiertos y serán almacenados en un datalake, de donde es posible consumir esta información luego por medio de motores de consulta, tendríamos un esquema de este tipo:



Dentro de S3 vamos a crear tres zonas principales:

- **[Zona raw](#)**: Donde llevaremos los datos crudos desde su fuente
- **[Zona Trusted](#)**: Donde se encontrarán datos de calidad y validación
- **[Zona refined](#)**: Se encontrarán los datos que contienen un resultado final que pueden ser utilizado para informes o visualizaciones.

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	<a href="#">trusted/</a>	Folder	-
<input type="checkbox"/>	<a href="#">refined/</a>	Folder	-
<input type="checkbox"/>	<a href="#">raw/</a>	Folder	-

Ingesta de los datos:

Al revisar la fuente original de los datos, podemos observar que estos pueden ser consumidos mediante una API, por lo que contamos con varias opciones para subir la información a S3

- 1- Configurar en el AWS CLI de nuestra maquina las llaves y mediante un script y Boto3 ingestar la data directamente a nuestro bucket destino en S3 [\[código\]](#)

```
def upload_to_s3(local_file, bucket, s3_key):
    # Creamos el cliente a S3
    s3 = boto3.client('s3')

    try:
        #UPLOAD del archivo a S3
        s3.upload_file(local_file, bucket, s3_key)
        print(f'Successfully uploaded {local_file} to {bucket}/{s3_key}')
    except FileNotFoundError:
        print(f'The file {local_file} was not found')
    except NoCredentialsError:
        print('Credentials not available')

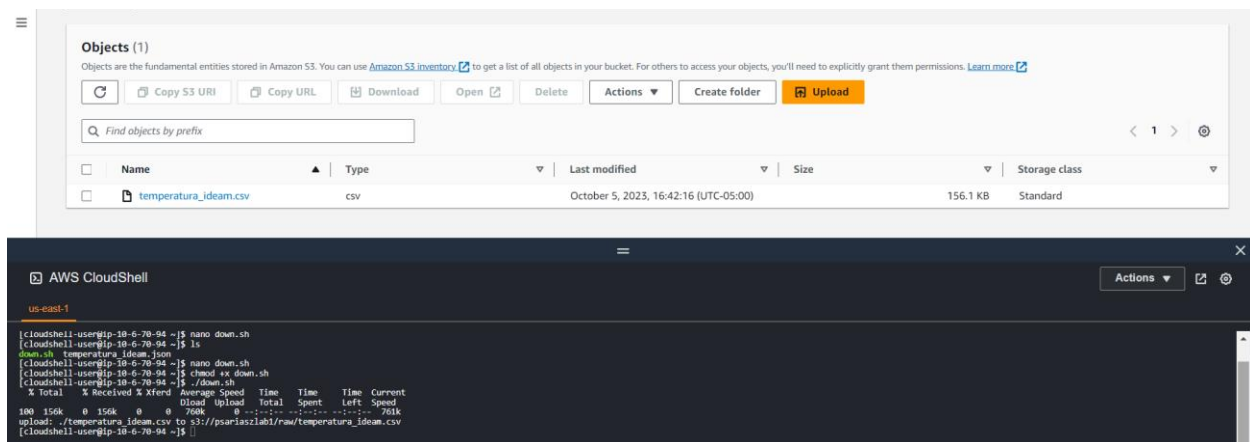
local_file_path = '/temperatura_ideam.json'
s3_bucket_name = 'psariaszlab1'
s3_key = 'raw/temperatura_ideam.json'

upload_to_s3(local_file_path, s3_bucket_name, s3_key)
```

- 2- Llevar a un script en bash que permita ingestar los datos directamente desde la fuente a un bucket en la Zona raw en S3. [\[código\]](#)

```
$ down.sh
1 # Comando para descargar el archivo
2 curl -o temperatura_ideam.csv https://www.datos.gov.co/resource/sbwg-7ju4.csv
3
4 # Comando para cargar el archivo en S3
5 aws s3 cp temperatura_ideam.csv "s3://psariaszlab1/raw/"
```

Posteriormente como estamos trabajando en una cuenta de AWS Academy y no contamos con los datos suficientes para configurar el AWS CLI desde nuestra maquina en local, se decide utilizar el cloud Shell que se tiene acceso dentro de AWS para ejecutar el archivo bash. Como se puede ver a continuación:



Dado el caso que haya alguna actualización de los datos desde la fuente solamente sería ejecturar este script desde la terminal y allí queda almacenado:

```

AWS CloudShell

us-east-1

[cloudshell-user@ip-10-6-70-94 ~]$ nano down.sh
[cloudshell-user@ip-10-6-70-94 ~]$ ls
down.sh  temperatura_ideam.json
[cloudshell-user@ip-10-6-70-94 ~]$ nano down.sh
[cloudshell-user@ip-10-6-70-94 ~]$ chmod +x down.sh
[cloudshell-user@ip-10-6-70-94 ~]$ ./down.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 156k    0 156k    0    0   760k    0 --:--:-- --:--:-- --:--:--  761k
upload: ./temperatura_ideam.csv to s3://psariaszlab1/raw/temperatura_ideam.csv
[cloudshell-user@ip-10-6-70-94 ~]$ ls
down.sh  temperatura_ideam.csv  temperatura_ideam.json
[cloudshell-user@ip-10-6-70-94 ~]$

```

## Catalogación

Con ayuda de Glue creamos un crawler que tome desde la zona raw en el S3 los datos allí ingestados y le realizamos una catalogación:

## cambio\_climatico

Last updated (UTC)  
October 5, 2023 at 21:45:51

Run crawler

Edit

Delete

## Crawler properties

Name cambio_climatico	IAM role <a href="#">LabRole</a>	Database trabajo_1	State RUNNING
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

► Advanced settings

[Crawler runs](#) | [Schedule](#) | [Data sources](#) | [Classifiers](#) | [Tags](#)

## Crawler runs (1)

The list of crawler runs for this crawler.



Stop run

[View CloudWatch logs](#)[View run details](#)

Q Filter data

Filter by a date and time range

&lt; 1 &gt; ⚙

	Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
<input type="radio"/>	October 5, 2023 at 21:45:45	October 5, 2023 at 21:46:44	59 s	Completed	-	-

Podemos observar que nos creó la siguiente tabla, con el siguiente schema:

Table overview

Data quality New

Table details

Advanced properties

Name

raw

Description

-

Database

trabajo\_1

Classification

CSV

Location

s3://psariaslab1/raw/

Connection

-

Deprecated

-

Last updated

October 5, 2023 at 21:46:44

Input format

org.apache.hadoop.mapred.TextInputFormat

Output format

org.apache.hadoop.hive.q1o.HiveIgnoreKeyTextOutputFormat

Serde serialization lib

org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Schema

Partitions

Indexes

Schema (12)

View and manage the table schema.

Q Filter schemas

Edit schema as JSON

Edit schema

< 1 > ⌕

#

Column name

Data type

Partition key

Comment

1

codigoestacion

bigint

-

-

2

codigo sensor

bigint

-

-

3

fechaobservacion

string

-

-

4

valorobservado

double

-

-

5

nombrestacion

string

-

-

6

departamento

string

-

-

7

municipio

string

-

-

8

zonahidrografica

string

-

-

9

latitud

double

-

-

10

longitud

double

-

-

11

descripcion sensor

string

-

-

12

unidadmedida

string

-

-



Podemos empezar a sospechar de nuestro schema, ya que este no está permitiendo que si los datos no cumplen la condición fde tipo de dato con la que fueron creados por el catalogador, estos no serán visibles, por lo que pasaremos a cambiarlos todos a string, ya que revisamos que desde nuestra fuente este es el tipo de dato con el que fueron publicados.

Schema						
Partitions						
Indexes						
Schema (12)						
View and manage the table schema.						
Filter schemas						
< 1 > ⚙						
#	Column name	Data type	Partition key	Comment		
1	codigoestacion	string	-	-		
2	codigosensor	string	-	-		
3	fechaobservacion	string	-	-		
4	valorobservado	string	-	-		
5	nombrestacion	string	-	-		
6	departamento	string	-	-		
7	municipio	string	-	-		
8	zonahidrografica	string	-	-		
9	latitud	string	-	-		
10	longitud	string	-	-		
11	descripcionsensor	string	-	-		
12	unidadmedida	string	-	-		

Sin embargo, esto no sería una buena práctica por lo que se busca una alternativa para que no tengamos que dejar todos los tipos de datos erradamente, así que se crea un classifier de la siguiente forma:

cambio\_climatico\_csv\_op2

Last updated (UTC)

October 5, 2023 at 22:28:41

↻

Edit

Delete

Classifier properties

Name

cambio\_climatico\_csv\_op2

Allow single column

False

CSV Serde

None

Contains header

Detect headings

Header

-

Creation time

October 5, 2023 at 22:23:26

Delimiter

,

Disable value trimming

True

Quote symbol

"

Last updated

October 5, 2023 at 22:23:26

Version

1

Custom datatypes

-

Schema

Partitions

Indexes

Schema (12)

View and manage the table schema.

🔍

Filter schemas

#	Column name	Data type	Partitions
1	codigoestacion	bigint	-
2	codigosensor	bigint	-
3	fechaobservacion	string	-
4	valorobservado	double	-
5	nombrestacion	string	-
6	departamento	string	-
7	municipio	string	-
8	zonahidrografica	string	-
9	latitud	double	-
10	longitud	double	-
11	descripcionsensor	string	-
12	unidadmedida	string	-

Y ya con esta nueva catalogación revisemos como se ve en Athena, realizando una consulta de SQL sencilla:

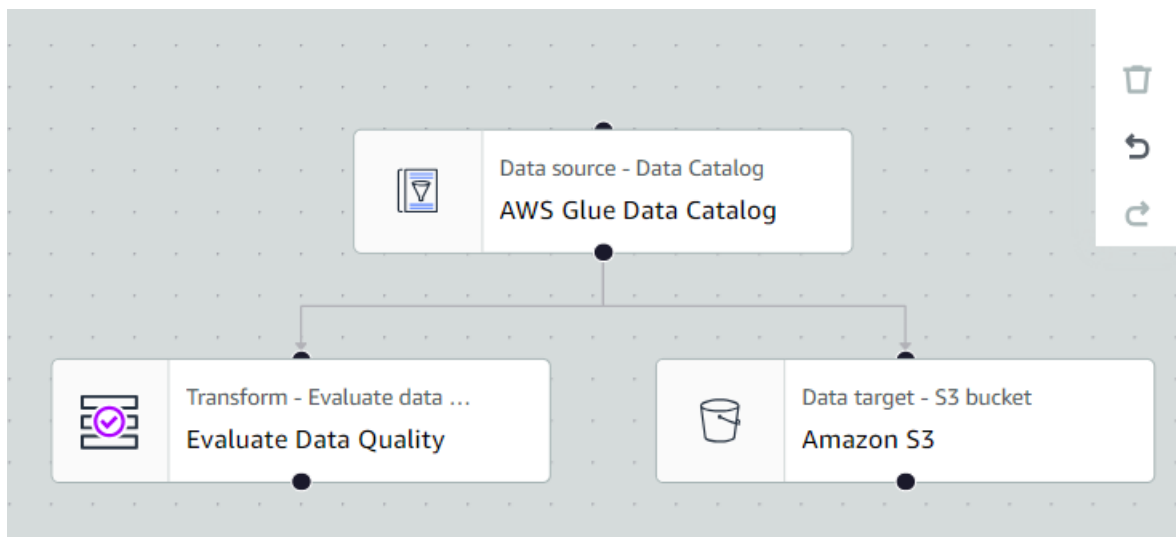
```
1 SELECT * FROM "AwsDataCatalog"."trabajo_1"."op2raw" limit 10;
```

#	codigoestacion	codigosensor	fechaobservacion	valorobservado	nombrestacion	de
1	23125170	68	2006-09-21T12:00:00.000	15.9	SAN CAYETANO - AUT	CU
2	16025501	68	2018-12-02T09:20:00.000	23.02354	CUCUTILLA - AUT	NO
3	52055501	68	2016-06-25T04:05:00.000	16.3	OSPINA PEREZ - AUT	NA
4	52055210	68	2014-06-02T08:00:00.000	13.3	BOTANA - AUT	NA
5	26155502	68	2019-07-27T00:05:00.000	20.45158	CENICAFE - AUT	CA
6	21208480	68	2012-03-16T15:15:00.000	19.1	KENNEDY - FOPAE	BO
7	21208480	68	2016-01-17T23:55:00.000	16.6	KENNEDY - FOPAE	BO
8	24035390	68	2015-02-24T00:00:00.000	2.9	PARAMO ALMORZADERO - AUT	SA
9	21205519	68	2019-04-20T03:50:00.000	11.072	SAN FORTUNATO - AUT	CU
10	23055501	68	2017-10-10T22:50:00.000	19.13098	SANTA HELENA - AUT	CA

Perfecto, ya podemos observar que nuestros datos se pueden visualizar completamente gracias al classifier aplicado, que corrigió de una forma adecuada nuestro schema.

## ETL GLUE → S3

Ahora bien, luego del proceso anterior vamos a realizar un proceso de ETL hacia la zona Trusted. Para esto vamos a utilizar **ETL Jobs** de **AWS Glue**, de la siguiente forma:





Donde el nodo de **Evaluate Data Quality** es que al menos la columna **valorobservado** sea de tipo **"Double"**. [\[código\]](#)

```
Script (Locked) Info
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from awsglue.transforms import EvaluateDataQuality
8
9 args = getResolvedOptions(sys.argv, ["JOB_NAME"])
10 sc = SparkContext()
11 glueContext = GlueContext(sc)
12 spark = glueContext.spark_session
13 job = Job(glueContext)
14 job.init(args["JOB_NAME"], args)
15
16 # Script generated for node AWS Glue Data Catalog
17 AWSGlueDataCatalog_node1696546497201 = glueContext.create_dynamic_frame.from_catalog(
18     database="trabajo_1",
19     table_name="op2raw",
20     transformation_ctx="AWSGlueDataCatalog_node1696546497201",
21 )
22
23 # Script generated for node Evaluate Data Quality
24 EvaluateDataQuality_node1696546509793_ruleset = """
25 # Example rules: Completeness "colA" between 0.4 and 0.8, ColumnCount > 10
26 Rules = [
27     ColumnDataType "valorobservado" = "double"
28 ]
29 """
30
31 EvaluateDataQuality_node1696546509793 = EvaluateDataQuality().process_rows(
32     frame=AWSGlueDataCatalog_node1696546497201,
33     ruleset=EvaluateDataQuality_node1696546509793_ruleset,
34     publishing_options={
35         "dataQualityEvaluationContext": "EvaluateDataQuality_node1696546509793",
36         "enableDataQualityCloudWatchMetrics": True,
37         "enableDataQualityResultsPublishing": True,
38     },
39     additional_options={"performanceTuning.caching": "CACHE_NOTHING"},
40 )
41
42 # Script generated for node Amazon S3
43 AmazonS3_node1696546512586 = glueContext.write_dynamic_frame.from_options(
44     frame=AWSGlueDataCatalog_node1696546497201,
45     connection_type="s3",
46     format="glueparquet",
47     connection_options={"path": "s3://psariaszlab1/trusted/", "partitionKeys": []},
48     format_options={"compression": "snappy"},
49     transformation_ctx="AmazonS3_node1696546512586",
50 )
51
52 job.commit()
```

Este código queda directamente guardado en el S3:

Amazon S3 > Buckets > [aws-glue-assets-300013369769-us-east-1](#) > scripts/

scripts/ Copy S3 URI

Objects Properties

**Objects (4)**  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	new_job_clima-csv.py	py	October 5, 2023, 18:08:50 (UTC-05:00)	1.8 KB	Standard
<input checked="" type="checkbox"/>	new_job_clima.py	py	October 5, 2023, 18:02:08 (UTC-05:00)	1.8 KB	Standard

Y si nos dirigimos ahora hacia nuestra zona Trusted dentro del S3 podemos ver ahora nuestro archivo con un proceso de calidad de datos:

Amazon S3 > Buckets > psariaszlab1 > trusted/

trusted/ Copy S3 URI

**Objects** | Properties

**Objects (2)**  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Refresh Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

< 1 > Settings

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	run-1696547086350-part-block-0-r-00000-snappy.parquet	parquet	October 5, 2023, 18:04:52 (UTC-05:00)	36.3 KB	Standard
<input type="checkbox"/>	run-1696547451782-part-r-00000.snappy	snappy	October 5, 2023, 18:10:53 (UTC-05:00)	46.3 KB	Standard

Para fines prácticos trabajaremos con el archivo con extensión parquet. El cual nos permite mantener nuestra data en un formato más liviano.

## Consulta realizada en Athena

De los datos catalogados en Glue, se realiza la siguiente consulta:

✓ municipios\_mas\_cali... ⋮

```
1 SELECT municipio, valorobservado,unidadmedida
2 FROM "AwsDataCatalog"."trabajo_1"."op2raw"
3 WHERE valorobservado >= 34;
```

Esto con el fin de obtener los municipios donde alguna estación del IDEAM tomo una temperatura mayor o igual a 34°C, y este fue el resultado:

#	municipio	valor observado	unidad medida
1	TÁMESIS	36.15168	°C
2	ARIGUANÍ (EL DIFICIL)	36.5	°C
3	PATÍA (EL BORDO)	35.9	°C
4	AMBALEMA	35.7	°C
5	BOGOTA, D.C	39.2	°C
6	VALLEDUPAR	35.8	°C

## AMAZON EMR

Ahora vamos a crear un cluster administrado para ejecutar el trabajo con esta data en un marco tipo Big Data.

A continuación, se muestran las aplicaciones que van a ser orquestadas por mi EMR:

almacenamiento\_cambio\_climatico

Amazon EMR release [Info](#)

A release contains a set of applications which can be installed on your cluster.

emr-6.13.0

Application bundle

Spark

Core Hadoop

Flink

HBase

Presto

Trino

Custom

☒ Flink 1.17.0

☒ HCatalog 3.1.3

☒ Hue 4.11.0

☒ Livy 0.7.1

☐ Phoenix 5.1.3

☒ Spark 3.4.1

☐ Tez 0.10.2

☒ ZooKeeper 3.5.10

☐ Ganglia 3.7.2

☒ Hadoop 3.3.3

☒ JupyterEnterpriseGateway 2.6.0

☐ MXNet 1.9.1

☐ Pig 0.17.0

☒ Sqoop 1.4.7

☐ Trino 414

☐ HBase 2.4.17

☒ Hive 3.1.3

☒ JupyterHub 1.5.0

☐ Oozie 5.2.1

☐ Presto 0.281

☐ TensorFlow 2.11.0

☒ Zeppelin 0.10.1

AWS Glue Data Catalog settings

Use the AWS Glue Data Catalog to provide an external metastore for your application.

☒ Use for Hive table metadata

☒ Use for Spark table metadata

Operating system options [Info](#)

☒ Amazon Linux release

Summary [Info](#)

Name and applications

Name

almacenamiento\_cambio\_climatico

Amazon EMR release

emr-6.13.0

Application bundle

Custom (Flink 1.17.0, HCatalog 3.1.3, Hadoop 3.3.3, Hive 3.1.3, Hue 4.11.0, JupyterEnter...)

Amazon Linux release

2.0.20230808.0

Cluster configuration

Instance groups

Primary (m4.xlarge), Core (m4.xlarge), Task (m4.xlarge)

Cluster scaling and provisioning option

Cancel

Clone cluster

## Security configuration and EC2 key pair - optional [Info](#)

### Security configuration

Select your cluster encryption, authentication, authorization, and instance metadata service settings.



[Browse](#)

[Create security configuration](#)

### Amazon EC2 key pair for SSH to the cluster [Info](#)



[Browse](#)

[Create key pair](#)

[Amazon EMR](#) > EMR on EC2: Clusters

### Clusters (5) [Info](#)



[View details](#)

[Terminate](#)

[Clone](#)

[Create cluster](#)

[Filter clusters by status](#)

[Filter clusters by creation date-time](#)

[< 1 >](#)

<input type="checkbox"/>	<input type="checkbox"/>	Cluster ID	Cluster name	Status	Creation time (UTC-05:00)	Elapsed time
<input type="checkbox"/>		j-2Q10R1HXJOCW	almacenamiento_cambio_climatico	Starting Preparing cluster	October 05, 2023, 19:01	9 seconds

Ahora bien, las aplicaciones seleccionadas en la parte anterior, algunas de ellas (Como Hive o Hadoop) podremos utilizarlas mediante un UI con la aplicación HUE y también seleccionada:

[Spark History Server UI](#)

### Application UIs on the primary node

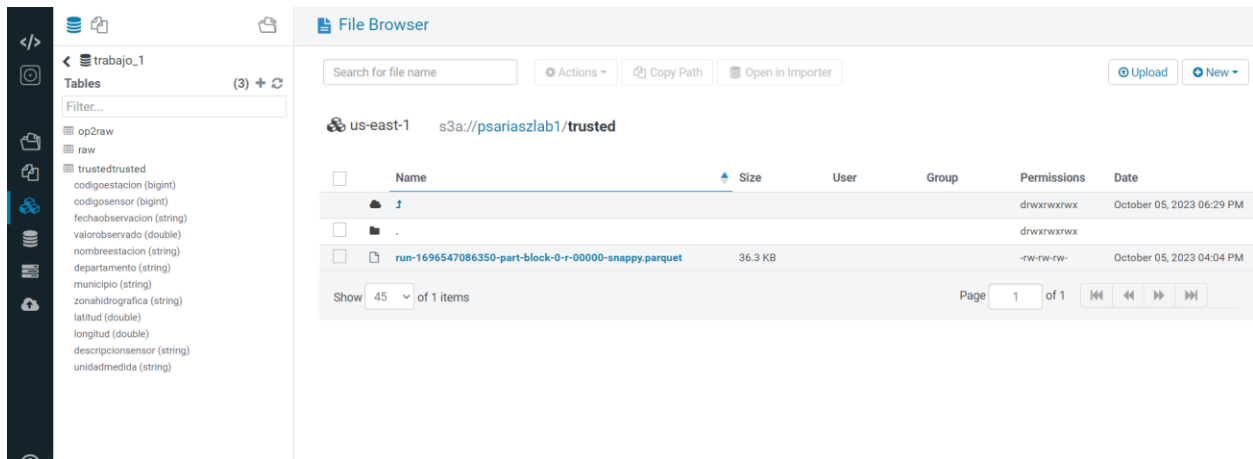
These require SSH tunneling to be enabled.

[Enable an SSH connection](#)

Application	UI URL <a href="#">↗</a>
HDFS Name Node	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:9870/">http://ec2-35-171-85-144.compute-1.amazonaws.com:9870/</a>
Hue	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:8888/">http://ec2-35-171-85-144.compute-1.amazonaws.com:8888/</a>
JupyterHub	<a href="https://ec2-35-171-85-144.compute-1.amazonaws.com:9443/">https://ec2-35-171-85-144.compute-1.amazonaws.com:9443/</a>
Livy	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:8998/">http://ec2-35-171-85-144.compute-1.amazonaws.com:8998/</a>
Resource Manager	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:8088/">http://ec2-35-171-85-144.compute-1.amazonaws.com:8088/</a>
Spark History Server	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:18080/">http://ec2-35-171-85-144.compute-1.amazonaws.com:18080/</a>
Zeppelin	<a href="http://ec2-35-171-85-144.compute-1.amazonaws.com:8890/">http://ec2-35-171-85-144.compute-1.amazonaws.com:8890/</a>

### Application UIs on the core and task nodes

Cuando entramos a este UI URL que nos proporciona el EMR creado, podemos observar que en dentro de HUE tenemos una entrada directa a S3:



## HDFS

Inicialmente dentro de la máquina virtual principal (EC2) orquestada por el EMR, vamos a conectarnos a la terminal y desde allí crear un nuevo directorio en HDFS, con el nombre: Trabajo\_1

```
[root@ip-172-31-46-194 conf]# hdfs dfs -ls /
Found 4 items
drwxr-xr-x - hdfs hdfsadmingroup 0 2023-10-06 00:39 /apps
drwxrwxrwt - hdfs hdfsadmingroup 0 2023-10-06 00:39 /tmp
drwxr-xr-x - hdfs hdfsadmingroup 0 2023-10-06 00:39 /user
drwxr-xr-x - hdfs hdfsadmingroup 0 2023-10-06 00:39 /var
[root@ip-172-31-46-194 conf]# hdfs dfs -ls /user
Found 9 items
drwxrwxrwx - hadoop hdfsadmingroup 0 2023-10-06 01:17 /user/hadoop
drwxr-xr-x - mapred mapred 0 2023-10-06 00:39 /user/history
drwxrwxrwx - hdfs hdfsadmingroup 0 2023-10-06 00:39 /user/hive
drwxrwxrwx - hue hue 0 2023-10-06 00:39 /user/hue
drwxrwxrwx - livy livy 0 2023-10-06 00:39 /user/livy
drwxrwxrwx - oozie oozie 0 2023-10-06 00:39 /user/oozie
drwxrwxrwx - root hdfsadmingroup 0 2023-10-06 00:39 /user/root
drwxrwxrwx - spark spark 0 2023-10-06 00:39 /user/spark
drwxrwxrwx - zeppelin hdfsadmingroup 0 2023-10-06 00:39 /user/zeppelin
[root@ip-172-31-46-194 conf]# hdfs dfs -mkdir /user/hadoop/trabajo_1
[root@ip-172-31-46-194 conf]# hdfs dfs -mkdir /user/hadoop/trabajo_1
[root@ip-172-31-46-194 conf]#
```

Dentro de esta misma terminal voy a realizar una copia de los datos que ingesten en la zona raw del S3 a la carpeta trabajo\_1 creada en HDFS (Recordemos que los datos almacenados allí son temporales).

```
[root@ip-172-31-46-194 conf]# hadoop distcp s3://psariaszlab1/raw/temperatura_ideam.csv /user/hadoop/trabajo_1
2023-10-06 01:48:55,418 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, de
append=false, useDiff=false, useRdiff=false, fromSnapshot=null, toSnapshot=null, skipCRC=false, blocking=true, m
strategy='uniformsize', preserveStatus=[], atomicWorkPath=null, logPath=null, sourceFileListing=null, sourcePaths
=/user/hadoop/trabajo_1, filtersFile='null', blocksPerChunk=0, copyBufferSize=8192, verboseLog=false, directWrit
1/raw/temperatura_ideam.csv], targetPathExists=true, preserveRawXattrs=false
2023-10-06 01:48:55,815 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at ip-172-
2023-10-06 01:48:56,004 INFO client.NHSPRow: Connecting to Application History server at ip-172-31-46-194 ec2-
```

Aquí podemos ver dentro de HUE, cuando entramos al directorio de Hadoop nuestro directorio trabajo\_1 con la data que copiamos directamente desde nuestro S3. (Nota importante, aquí también podemos crear directorio y subir archivos mediante los botones que se encuentran en el UI).

File Browser

Search for file name

Actions

Copy Path

Open in Importer

Upload

New

Home /user/hadoop/trabajo\_1

	Name	Size	User	Group	Permissions	Date
	↑		hadoop	hdfsadmingroup	drwxrwxrwx	October 05, 2023 06:40 PM
	↓		root	hdfsadmingroup	drwxr-xr-x	October 05, 2023 06:49 PM
	temperatura_ideam.csv	156.1 KB	root	hdfsadmingroup	-rw-r--	October 05, 2023 06:49 PM

Show 45 of 1 items

Page 1 of 1

File Browser

Back

Home

Page 1 to 40 of 40

Edit file

Refresh

View as binary

Download

Last modified 10/05/2023 8:49 PM

User root

Group hdfsadmingroup

Size 156.11 KB

Mode 100644

/ user/ hadoop/ trabajo\_1/ temperatura\_ideam.csv

```
"codigoestacion","codigosensor","fechaobservacion","valorobservado","nombrestacion","departamento","municipio","zonahidrografi
ca","latitud","longitud","descripcionsensor","unidadmedida"
"0023125170","0068","2006-09-21T12:00:00.000","15.9","SAN CAYETANO - AUT","CUNDINAMARCA","SAN CAYETANO","MEDIO MAGDALENA","5.33
333333","-74.016666667","Temp Aire 2 m","°C"
"0016025501","0068","2018-12-02T09:20:00.000","23.02354","CUCUTILLA - AUT","NORTE DE SANTANDER","CUCUTILLA","CATATUMBO","7.519
455556","-72.79790278","Temp Aire 2 m","°C"
"0052055501","0068","2016-06-25T04:05:00.000","16.3","OSPINA PEREZ - AUT","NARIÑO","CONSACÁ","PATÍA","1.254722222","-77.487777
78","Temp Aire 2 m","°C"
"0052055210","0068","2014-06-02T08:00:00.000","13.3","BOTANA - AUT","NARIÑO","PASTO","PATÍA","1.16","-77.27880556","Temp Aire 2
m","°C"
"0026155502","0068","2019-07-27T00:05:00.000","20.45158","CENICAFE - AUT","CALDAS","MANIZALES","CAUCA","4.991111111","-75.5974
9722","Temp Aire 2 m","°C"
"0021208480","0068","2012-03-16T15:15:00.000","19.1","KENNEDY - FOPAE","BOGOTA D.C.","BOGOTA, D.C","ALTO MAGDALENA","4.633","-7
4.15","Temp Aire 2 m","°C"
"0021208480","0068","2016-01-17T23:55:00.000","16.6","KENNEDY - FOPAE","BOGOTA D.C.","BOGOTA, D.C","ALTO MAGDALENA","4.633","-7
4.15","Temp Aire 2 m","°C"
"0024035390","0068","2015-02-24T00:00:00.000","2.9","PARAMO ALMORZADERO - AUT","SANTANDER","CERRITO","SOGAMOSO","6.94538888
9","-72.69633333","Temp Aire 2 m","°C"
"0021205519","0068","2019-04-20T03:50:00.000","11.072","SAN FORTUNATO - AUT","CUNDINAMARCA","SIBATE","ALTO MAGDALENA","4.45
2","-74.272","Temp Aire 2 m","°C"
"0023055501","0068","2017-10-10T22:50:00.000","19.13098","SANTA HELENA - AUT","CALDAS","MARQUETALIA","MEDIO MAGDALENA","5.3174
97222","-74.99610833","Temp Aire 2 m","°C"
"0026175502","0068","2017-09-05T09:20:00.000","25.59289","LA CRISTALINA - AUT","ANTIOQUIA","TÁMESIS","CAUCA","5.7","-75.659997
22","Temp Aire 2 m","°C"
"0024015501","0068","2019-07-26T10:05:00.000","22.53393","BERTHA - AUT","BOYACÁ","MONIQUEIRÁ","SOGAMOSO","5.882777778","-73.573
```

## HIVE

De nuevo en nuestra terminal de nuestra máquina virtual principal podemos observar que existe otro directorio para lo que tiene que ver con HIVE. Por lo que si deseamos realizar una consulta con HIVE de los datos almacenados en HDFS vamos a realizar una copia con el siguiente comando como vemos a continuación:

```
[root@ip-172-31-46-194 conf]# hdfs dfs -cp hdfs:///user/hadoop/trabajo_1/temperatura_ideam.csv /user/hive/warehouse
[root@ip-172-31-46-194 conf]#
```

Ahora bien, para poder observar estos datos en nuestra consulta de SQL de HIVE podemos realizarlo mediante una consulta o a través de la herramienta Importer de la siguiente forma:

File Browser

Search for file name

Actions

Copy Path

Open in Importer

Home

/user/hive/warehouse

	Name	Size	User	Group	F
<input type="checkbox"/>	<a href="#">↑</a>		hdfs	hdfsadmingroup	d
<input type="checkbox"/>	<a href="#">.</a>		hdfs	hdfsadmingroup	d
<input checked="" type="checkbox"/>	<a href="#">temperatura_ideam.csv</a>	156.1 KB	root	hdfsadmingroup	-i

Show

45

of 1 items

Page

1

Seleccionamos que es estilo CSV y delimitados por coma.

Pick data from file /user/hive/warehouse/temperatura\_ideam.csv

Move it to table default.temperatura\_ideam

SOURCE

Type

Remote File

Path

/user/hive/warehouse/temperatura\_ideam.csv

..

i

FORMAT

File Type

CSV File

Field Separator

Comma (,)

Record Separator

New line

Quote Character

Double Quote

☒ Has Header

PREVIEW

codigoestacion	codigosensor	fechaobservacion	valorobservado	nombrestacion	departamento	municipio
0023125170	0068	2006-09-21T12:00:00....	15.9	SAN CAYETANO - AUT	CUNDINAMARCA	SAN CAYETANO
0016025501	0068	2018-12-02T09:20:00....	23.02354	CUCUTILLA - AUT	NORTE DE SANTANDER	CUCUTILLA
0052055501	0068	2016-06-25T04:05:00....	16.3	OSPINA PEREZ - AUT	NARIÑO	CONSACÁ

Next

Revisamos que el schema que nos proporciona el importer es correcto:

## PROPERTIES

Format Text

Extras ≡



Partitions + Add partition

## FIELDS


Name	<input type="text" value="codigoestacion"/>	Type	<span>bigint</span>	<input type="checkbox"/>	0023125170	0016025501
Name	<input type="text" value="codigosensor"/>	Type	<span>bigint</span>	<input type="checkbox"/>	0068	0068
Name	<input type="text" value="fechaobservacion"/>	Type	<span>timestamp</span>	<input type="checkbox"/>	2006-09-21T12:00:00...	2018-12-02T09:20:00...
Name	<input type="text" value="valorobservado"/>	Type	<span>double</span>	<input type="checkbox"/>	15.9	23.02354
Name	<input type="text" value="nombrestacion"/>	Type	<span>string</span>	<input type="checkbox"/>	SAN CAYETANO - AUT	CUCUTILLA - AUT
Name	<input type="text" value="departamento"/>	Type	<span>string</span>	<input type="checkbox"/>	CUNDINAMARCA	NORTE DE SANTANDER

Back Submit


Ahora bien, la opción mostrada arriba es mediante el UI. Pero qué pasa si quisiéramos crear directamente la tabla en HIVE con una consulta SQL de los datos almacenados en Hadoop(HDFS) sin necesidad de pasarlos al directorio de HIVE. Utilizamos la siguiente sentencia de código: [\[código\]](#)





trabajo\_1

Tables (4) + 

Filter...

 op2raw

 raw

 temperaturas\_ideam

idestacion (string)

sensor (string)

fecha (string)

valorobservado (string)

nombrestacion (string)

departamento (string)

municipio (string)


zonahidro (string)


latitud (string)

longitud (string)




descripcion (string)



unidad (string)

 trustedtrusted

 Hive


↺ Add a name... Add a description...



0.64s trabajo\_1  

```
2 CREATE EXTERNAL TABLE temperaturas_ideam (  
3     idestacion BIGINT,  
4     sensor BIGINT,  
5     fecha TIMESTAMP,  
6     valorobservado DOUBLE,  
7     nombrestacion STRING,  
8     departamento STRING,  
9     municipio STRING,  
10    zonahidro STRING,  
11    latitud DOUBLE,  
12    longitud DOUBLE,  
13    descripcion STRING,  
14    unidad STRING  
15 )  
16 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
17 WITH SERDEPROPERTIES (  
18     "separatorChar" = ",",  
19     "quoteChar" = "\"",  
20     "escapeChar" = "\\\"  
21 )  
22 LOCATION '/user/hadoop/trabajo_1/';
```

```
    "separatorChar" = ",",  
    "quoteChar" = "\"",  
    "escapeChar" = "\\\"  
)  
LOCATION '/user/hadoop/trabajo_1/'  
INFO : Starting task [Stage-0:DDL] in serial mode  
INFO : Completed executing command (queryId=hive_20221006023152_6441a56f4_0000_4a00_a026_1e117...
```

 Success.



Creamos exitosamente nuestra tabla temperaturas\_ideam:

The screenshot shows a Hive query interface. On the left, a sidebar lists tables under 'trabajo\_1', with 'temperaturas\_ideam' highlighted. The main area displays a query: `SELECT * FROM temperaturas_ideam`. Below the query, a log shows the command execution details. At the bottom, a table titled 'Results (100+)' displays the data from the query.

	temperaturas_ideam.idestacion	temperaturas_ideam.sensor	temperaturas_ide
1	"codigoestacion"	"codigosensor"	"fechaobservacion"
2	"0023125170"	"0068"	"2006-09-21T12:00"
3	"0016025501"	"0068"	"2018-12-02T09:20"
4	"0052055501"	"0068"	"2016-06-25T04:05"
5	"0052055210"	"0068"	"2014-06-02T08:00"
6	"0026155502"	"0068"	"2019-07-27T00:05"
7	"0021208480"	"0068"	"2012-03-16T15:15"

## CONSULTA EN HIVE A S3

En este caso de uso podemos ver, como podríamos realizar una consulta en Hive directamente a los datos que tenemos en nuestra zona Trusted en S3, creamos una tabla con su respectivo schema y el nombre **temperaturas\_ideam\_s3** [código]:

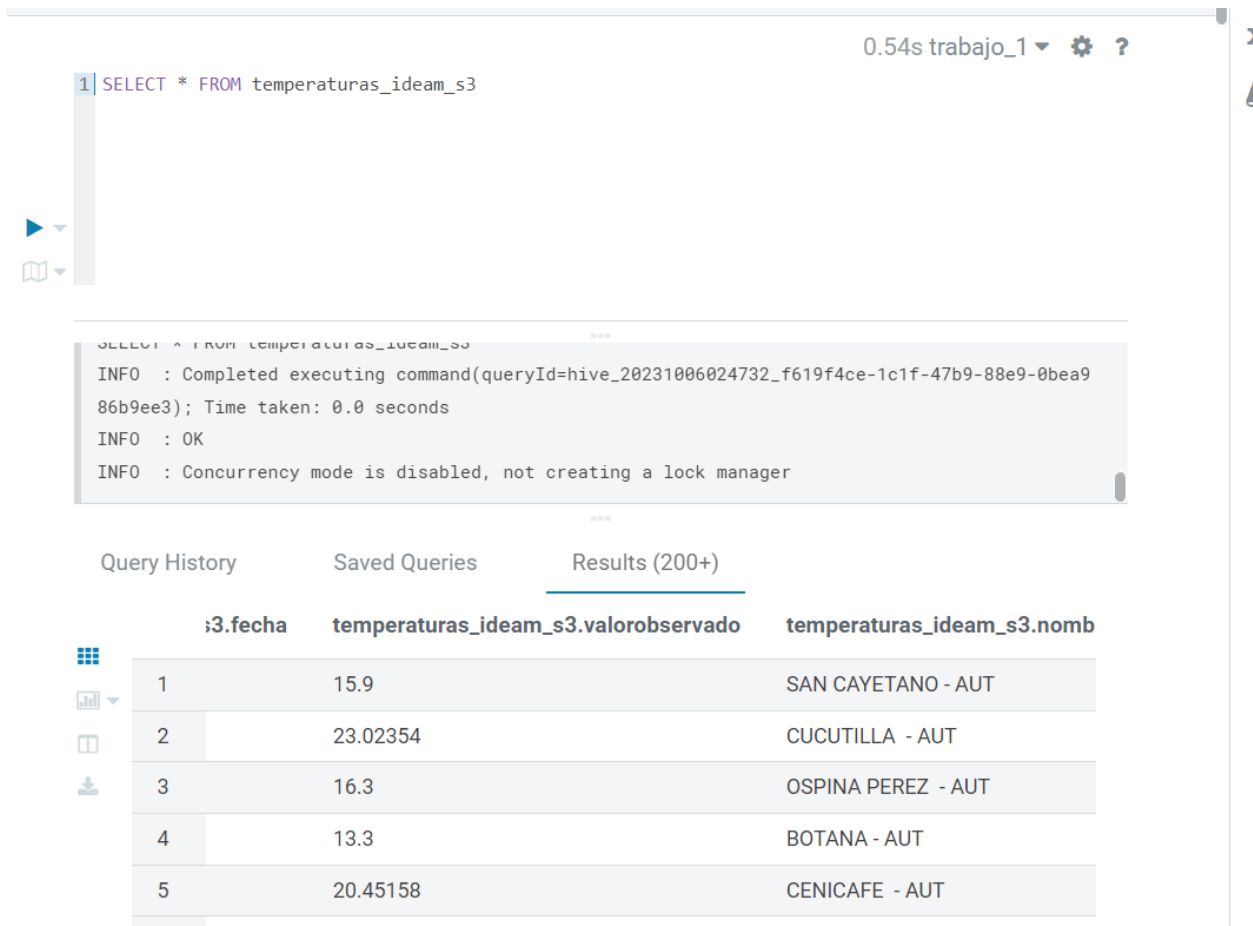
The screenshot shows a Hive query interface. On the left, a sidebar lists tables under 'trabajo\_1', with 'temperaturas\_ideam\_s3' highlighted. The main area displays a query to create an external table. Below the query, a log shows the command execution details, and a success message is displayed at the bottom.

```
1 use trabajo_1;
2 CREATE EXTERNAL TABLE temperaturas_ideam_s3 (
3   idestacion BIGINT,
4   sensor BIGINT,
5   fecha TIMESTAMP,
6   valorobservado DOUBLE,
7   nombrestacion STRING,
8   departamento STRING,
9   municipio STRING,
10  zonahidro STRING,
11  latitud DOUBLE,
12  longitud DOUBLE,
13  descripcion STRING,
14  unidad STRING
15 )
16 STORED AS PARQUET
17 LOCATION 's3://psariaszlabi/trusted'
```

INFO : Compiling command(queryId=hive\_20231006024613\_b3fa878f-c655-4023-867e-6abfe1394809):  
CREATE EXTERNAL TABLE temperaturas\_ideam\_s3 (  
 idestacion BIGINT,  
 sensor BIGINT,  
 fecha TIMESTAMP,  
 valorobservado DOUBLE,  
 nombrestacion STRING,  
 departamento STRING,  
 municipio STRING,  
 zonahidro STRING,  
 latitud DOUBLE,  
 longitud DOUBLE,  
 descripcion STRING,  
 unidad STRING  
)  
STORED AS PARQUET  
LOCATION 's3://psariaszlabi/trusted'

Success.

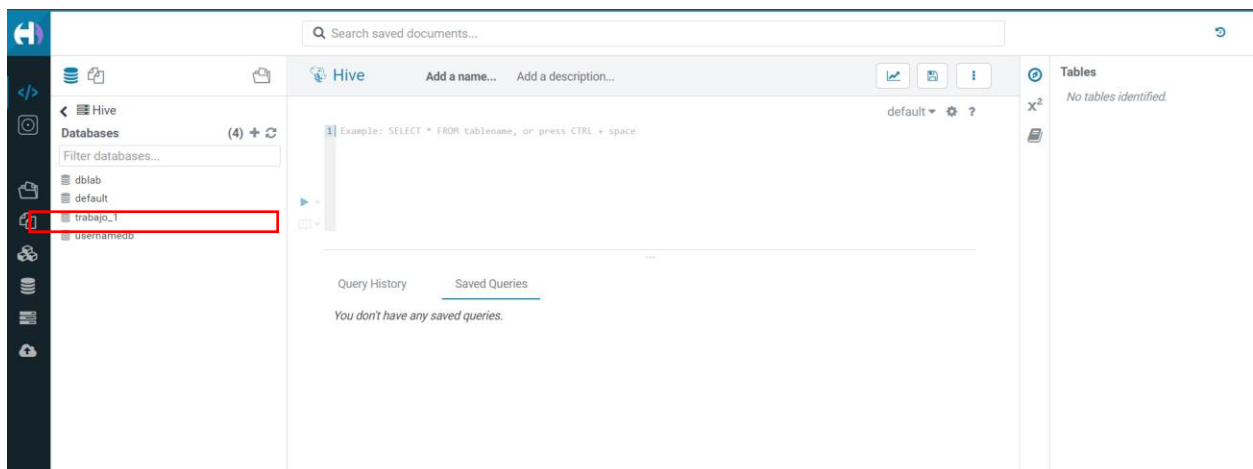
Hacemos una consulta a la tabla creada y podemos observar la data que teníamos en nuestra zona Trusted, dentro de Hive:



The screenshot shows the Hive console interface. At the top, a status bar indicates "0.54s trabajo\_1" with settings and help icons. The query editor contains the command: `1 | SELECT * FROM temperaturas_ideam_s3`. Below the editor, the execution log shows: `INFO : Completed executing command(queryId=hive_20231006024732_f619f4ce-1c1f-47b9-88e9-0bea986b9ee3); Time taken: 0.0 seconds`, `INFO : OK`, and `INFO : Concurrency mode is disabled, not creating a lock manager`. The results tab is active, displaying a table with 3 columns: `id.fecha`, `temperaturas_ideam_s3.valorobservado`, and `temperaturas_ideam_s3.nomb`. The table contains 5 rows of data.

	<code>id.fecha</code>	<code>temperaturas_ideam_s3.valorobservado</code>	<code>temperaturas_ideam_s3.nomb</code>
1		15.9	SAN CAYETANO - AUT
2		23.02354	CUCUTILLA - AUT
3		16.3	OSPINA PEREZ - AUT
4		13.3	BOTANA - AUT
5		20.45158	CENICAFE - AUT

También recordemos que al principio la data que paso por la zona de raw llega a Glue para ser catalogada, allí nuestros datos quedaron guardados en una base de datos llamada `trabajo_1`. Es decir, todo lo que llevemos a Glue podremos realizar consultas con Hive, como vemos a continuación: [\[código\]](#)



The screenshot shows the AWS Glue console. On the left, the "Databases" list shows `trabajo_1` highlighted with a red box. The main panel shows the Hive query editor with the example query: `Example: SELECT * FROM tablename, or press CTRL + space`. The "Tables" panel on the right indicates "No tables identified".




## JUPYTERHUB

Para estos puntos trabajaremos en un ambiente de datos basado en Spark con PySpark utilizando los jupyter notebooks. Recordemos que esta fue una de las aplicaciones que adquirimos en nuestro EMR. Por lo que nos dirigimos allí para abrir su URL y acceder (Usuario: jovyan, Clave: Jupyter). [\[código\]](#)

Application UIs on the primary node		Enable an SSH connection
These require SSH tunneling to be enabled.		
Application	UI URL	
HDFS Name Node	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:9870/">http://ec2-3-94-204-50.compute-1.amazonaws.com:9870/</a>	
Hue	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:8888/">http://ec2-3-94-204-50.compute-1.amazonaws.com:8888/</a>	
JupyterHub	<a href="https://ec2-3-94-204-50.compute-1.amazonaws.com:9443/">https://ec2-3-94-204-50.compute-1.amazonaws.com:9443/</a>	
Livy	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:8998/">http://ec2-3-94-204-50.compute-1.amazonaws.com:8998/</a>	
Resource Manager	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:8088/">http://ec2-3-94-204-50.compute-1.amazonaws.com:8088/</a>	
Spark History Server	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:18080/">http://ec2-3-94-204-50.compute-1.amazonaws.com:18080/</a>	
Zeppelin	<a href="http://ec2-3-94-204-50.compute-1.amazonaws.com:8890/">http://ec2-3-94-204-50.compute-1.amazonaws.com:8890/</a>	

Vamos a llamar a nuestro jupyter, con el nombre de temperatura\_ideam\_analisis, ya que esto es lo que deseamos hacer precisamente en este ambiente un análisis.

 temperatura\_ideam\_analisis

Last Checkpoint: in a few seconds (autosaved)

Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted PySpark

In [2]:

```
%%configure -f
{
  "conf":{
    "spark.pyspark.python": "python3",
    "spark.pyspark.virtualenv.enabled": "true",
    "spark.pyspark.virtualenv.type": "native",
    "spark.pyspark.virtualenv.bin.path": "/usr/bin/virtualenv"
  }}

```

Current session configs: {'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.virtualenv.enabled': 'true', 'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path': '/usr/bin/virtualenv'}, 'proxyUser': 'jovyan', 'kind': 'pyspark'}

No active sessions.

In [8]:

```
df = spark.read.option("header", True).csv("s3://psariaszlab1/raw/temperatura_ideam.csv", inferSchema=True)
```

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.

In [9]:

```
df.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.

codigoestacion	codigosensor	fechaobservacion	valorobservado	nombreestacion	departamento	municipio	zonahidrogr
rafica	latitud	longitud	descripcion	sensor	unidadmedida		

Jupyterhubtemperatura\_ideam\_analisisLast checkpoint a minute ago (unsaved changes)

FileEditViewInsertCellKernelWidgetsHelp

TrustedPySpark

In [25]: df.show(10)

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.

codigoestacion	codigo sensor	fechaobservacion	valor observado	nombrestacion	departamento	municipio	zonahidrog
rafica	latitud	longitud	descripcion sensor	unidad medida			
	23125170	68	2086-09-21 12:00:00	15.9	SAN CAYETANO - AUT	CUNDINAMARCA	SAN CAYETANO  MEDIO MAG
DALENA	5.333333333	-74.01666667	Temp Aire 2 m	°C			
	16025501	68	2018-12-02 09:20:00	23.02354	CUCUTILLA - AUT	NORTE DE SANTANDER	CUCUTILLA  CAT
ATUMBO	7.519455556	-72.79790278	Temp Aire 2 m	°C			
	52055501	68	2016-06-25 04:05:00	16.3	OSPINA PEREZ - AUT	HARIÑO	CONSACÁ
PATÍA	1.254722222	-77.48777778	Temp Aire 2 m	°C			
	52055210	68	2014-06-02 08:00:00	13.3	BOTANA - AUT	HARIÑO	PASTO
PATÍA	1.16	-77.27880556	Temp Aire 2 m	°C			
	26155502	68	2019-07-27 00:05:00	20.45158	CENICAFE - AUT	CALDAS	MANIZALES
CAUCA	4.991111111	-75.59749722	Temp Aire 2 m	°C			
	21208480	68	2012-03-16 15:15:00	19.1	KENNEDY - FOPEA	BOGOTA D.C.	BOGOTA, D.C  ALTO MAG
DALENA	4.633	-74.15	Temp Aire 2 m	°C			
	21208480	68	2016-01-17 23:55:00	16.6	KENNEDY - FOPEA	BOGOTA D.C.	BOGOTA, D.C  ALTO MAG
DALENA	4.633	-74.15	Temp Aire 2 m	°C			
	24035390	68	2015-02-24 00:00:00	2.9	PARAMO ALMORZADER...	SANTANDER	CERRITO  SO
GAMOSO	6.945388889	-72.69633333	Temp Aire 2 m	°C			
	21205519	68	2019-04-20 03:50:00	11.072	SAN FORTUNATO - AUT	CUNDINAMARCA	SIBATÉ  ALTO MAG
DALENA	4.452	-74.272	Temp Aire 2 m	°C			
	23055501	68	2017-10-10 22:50:00	19.13098	SANTA HELENA - AUT	CALDAS	MARQUETALIA  MEDIO MAG
DALENA	5.317497222	-74.99610833	Temp Aire 2 m	°C			

only showing top 10 rows

En esta imagen anterior podemos ver que estamos consumiendo nuestros datos directamente desde nuestro S3 y mediante PySpark, podemos visualizar nuestros diez primeros datos. A continuación, haremos una agrupación entre el promedio de valor observado de temperatura por departamento. Donde el promedio de temperatura mas bajo lo tiene Cundinamarca y la temperatura mas alta La Guajira.

```
In [10]: df.groupby('departamento').agg({'valorobservado': 'mean'}).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.

```
+-----+
| departamento | avg(valorobservado) |
+-----+
| CORDOBA      | 26.425 |
| CASANARE     | 21.934563333333333 |
| ANTIOQUIA    | 20.242767605633805 |
| BOLIVAR      | 28.4 |
| CUNDINAMARCA | 16.364248021978014 |
| NORTE DE SANTANDER | 19.9874839 |
| PUTUMAYO     | 24.14285714285714 |
| NARIÑO       | 16.74514676470588 |
| BOYACÁ       | 16.815409318181818 |
| GUAINÍA      | 23.2 |
| CAUCA        | 18.95000137254902 |
| LA GUAJIRA   | 28.657142857142862 |
| HUILA        | 19.047806315789476 |
| SUCRE        | 22.333333333333334 |
| QUINDÍO      | 20.79653689655172 |
| CESAR        | 26.872727272727275 |
| SANTANDER    | 17.159254193548385 |
| CAQUETA      | 22.985110714285714 |
| AMAZONAS     | 26.55 |
| CHOCO        | 22.655575 |
+-----+
```

only showing top 20 rows

Revisamos mediante un filtrado si encontramos datos faltantes en nuestro dataframe y cantidad de valores únicos por cada variable que lo componen:

```
In [11]: # Número de datos faltantes por columna
for column in df.columns:
    missing_count = df.filter(df[column].isnull()).count()
    print(f"Missing values in {column}: {missing_count}")
```

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.

```
Missing values in codigoestacion: 0
Missing values in codigosensor: 0
Missing values in fechaobservacion: 0
Missing values in valorobservado: 0
Missing values in nombrestacion: 0
Missing values in departamento: 0
Missing values in municipio: 0
Missing values in zonahidrografica: 0
Missing values in latitud: 0
Missing values in longitud: 0
Missing values in descripcionsensor: 0
Missing values in unidadmedida: 0
```

```
In [12]: for column in df.columns:
    print(f"Unique values in {column}: {df.select(column).distinct().count()}")
```

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.

```
Unique values in codigoestacion: 300
Unique values in codigosensor: 1
Unique values in fechaobservacion: 997
Unique values in valorobservado: 666
Unique values in nombrestacion: 299
Unique values in departamento: 35
Unique values in municipio: 226
Unique values in zonahidrografica: 29
Unique values in latitud: 295
Unique values in longitud: 293
Unique values in descripcionsensor: 1
Unique values in unidadmedida: 1
```

Por último, PySpark también nos permite hacer una descripción de los datos, mediante el método describe(). Dándonos datos útiles como la media, la desviación estándar o mínimos y máximos.

```
In [19]: df.describe().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.


```
+-----+-----+-----+-----+-----+-----+-----+-----+
|summary| codigoestacion|codigosensor| valorobservado| nombrestacion| departamento| municipio| zonahidrografica|
| latitud| longitud|descripcionsensor|unidadmedida|
+-----+-----+-----+-----+-----+-----+-----+-----+
| count| 1000| 1000| 1000| 1000| 1000| 1000| 1000|
| mean| 1.48645602567E8| 68.0|18.798422145999993| null| null| null| null|
| 4.987807572998997| -74.94852337826308| null| null|
| stddev| 4.941658493589811E8| 0.0|6.0871743047574345| null| null| null| null|
| 2.1086536665475064|1.3385316631454822| null| null|
| min| 11025501| 68| 0.0| ACEVEDO - AUT| AMAZONAS| ACANDÍ| <nil>
| -3.78| -81.731| Temp Aire 2 m| °C|
| max| 3505500061| 68| 39.2| ZETAQUIRA - AUT| VALLE DEL CAUCA| ZONA BANANERA| TAPAJE - DAGUA - ...
| 12.542| -67.932| Temp Aire 2 m| °C|
+-----+-----+-----+-----+-----+-----+-----+-----+
```


```
In [ ]:
```

## Redshift

Para trabajar con redshift primero realizaremos la creación de nuestro cluster, para poder realizar el modelado de nuestros datos de temperatura.

### Create cluster [Info](#)

 Looking for free trial? Try Redshift Serverless. First-time Redshift Serverless customers receive a \$300 credit to use in their account.

Launch Redshift Serverless 

#### Cluster configuration

**Cluster identifier**  
This is the unique key that identifies a cluster.

redshift-cluster-1

The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

**Choose the size of the cluster**

☒ I'll choose

☐ Help me choose

**Node type** [Info](#)  
Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.

dc2.large ▼

**Number of nodes**  
Enter the number of nodes that you need.


1

Range (1-32)

## Database configurations

### Admin user name

Enter a login ID for the admin user of your DB instance.

The name must be 1-128 alphanumeric characters, and it can't be a [reserved word](#) .

### ☐ Auto generate password

Amazon Redshift can generate a password for you, or you can specify your own password.

### Admin user password

Must be 8-64 characters long. Must contain at least one uppercase letter, one lowercase letter and one number. Can be any printable ASCII character except "/", "", or "@".

☒ Show password



### Associated IAM roles (2) [Info](#)

Set default ▼

Manage IAM roles ▼

Create, associate, or remove an IAM role. You can associate up to 50 IAM roles. You can also choose an IAM role and set it as the default for this cluster.

< 1 >

<input type="checkbox"/>	<a href="#">IAM roles</a>	Status	Role type
<input type="checkbox"/>	<a href="#">LabRole</a>	Not applied	--
<input type="checkbox"/>	<a href="#">myRedshiftRole</a>	Not applied	--

### Additional configurations ☒ Use defaults

These configurations are optional, and default settings have been defined to help you get started with your cluster. Turn off "Use defaults" to modify these settings now.

#### Network

Using **default VPC** (`vpc-0a6e2d500907afc97`) and **default** subnet.

#### Security

Using **default** (`sg-0ae4767230a7a9ac9`) cluster security group.

#### Configuration

Using **default.redshift-1.0** parameter group with no database encryption.

#### Backup

Automated snapshots are created about every eight hours or following every 5 GB per node of data changes, whichever comes first.

#### Maintenance

Using **current** maintenance track.

Cancel

Create cluster

### Query data using Redshift query editor

Use the query editor v2 to run queries in your Redshift cluster.

Query data

### Work with your client tools

You can connect to Amazon Redshift from your client tools, such as SQL clients, business intelligence (BI) tools, and extract, transform, load (ETL) tools, using JDBC or ODBC drivers.

Cluster

redshift-cluster-1

Copy JDBC URL

Copy ODBC URL

### Choose your JDBC or ODBC driver

Use JDBC or ODBC drivers to connect to Amazon Redshift from your client tools, such as SQL clients, BI tools, and ETL tools. We recommend using the new Amazon Redshift-specific drivers for better performance and scalability.

Driver

JDBC 4.2 without AWS SDK (.jar)

Download driver

Clusters (1) Info

Query data Actions Create cluster

Filter clusters by property or value

< 1 > ⚙

<input type="checkbox"/>	Cluster	Status	Cluster namespace	Availability Zone	Multi-AZ	Storage capacity us...
<input type="checkbox"/>	redshift-cluster-1 dc2.large   1 node   16 TB	Available	6bccd82f-b936-4854-...	us-east-1e	No	

Amazon Redshift > Clusters > redshift-cluster-1

## redshift-cluster-1

Actions

Edit

Add partner integration

Query data

Query in query editor

Query in query editor v2

### General information Info

Cluster identifier redshift-cluster-1	Status Available	Node type dc2.large	Endpoint redshift-cluster-1.cy7q3fiwsv9a.us-east-1.redshift.amazonaws.com:5439/dev
Custom domain name - new -	Date created October 08, 2023, 09:57 (UTC-05:00)	Number of nodes 1	JDBC URL jdbc:redshift://redshift-cluster-1.cy7q3fiwsv9a.us-east-1.redshift.amazonaws.com:5439/dev
Cluster ARN arn:aws:redshift:us-east-1:300013369769:namespace:6bccd82f-b936-4854-b5c5-b5eb86922bbd	Storage used -		ODBC URL Driver={Amazon Redshift (x64)}; Server=redshift-cluster-1.cy7q3fiwsv9a.us-east-1.redshift.amazonaws.com; Database=dev
Cluster configuration Production	Multi-AZ No		

Cluster performance

Query monitoring

Schedules

Maintenance

Properties

## Edit connection for redshift-cluster-1

Authentication [Learn more](#)

☐ Federated user

The principal tags of your IAM role or user must provide the connection details. You configure these tags in IAM or your identity provider (IdP).

☐ Temporary credentials

The query editor v2 generates a temporary password to connect to the database.

☒ Database user name and password

Provide a database user and password for the database that you are connecting to. The query editor v2 stores your credentials in AWS Secrets Manager on your behalf.

☐ AWS Secrets Manager

Choose a secret with credentials that are associated with the cluster or that you created in AWS Secrets Manager. Only secrets tagged with a key starting with 'Redshift' are listed.

Database

dev

The database name must be 1-64 characters. Valid characters are lowercase alphanumeric characters.

User name

awsuser

Password

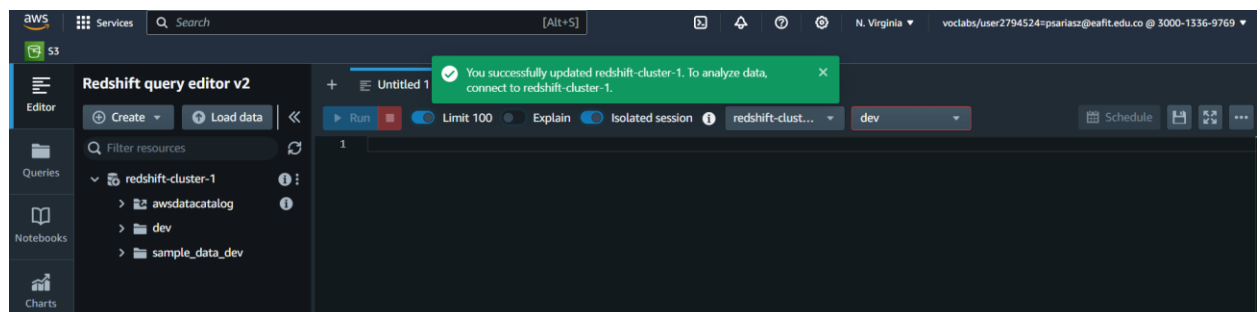
Maestria159\*

☒ Show password

Cancel

Save

© 2023, Amazon



Ya dentro de Redshift es importante tener a la mano nuestro ARN de nuestro rol: LabRole. Dado que gracias a este podremos crear tablas y realizar consultas en Redshift provenientes de S3.

The screenshot shows the AWS IAM console page for the 'LabRole'. The breadcrumb navigation at the top reads 'IAM > Roles > LabRole'. The page title is 'LabRole' with an 'Info' link. There are 'Delete' and 'Edit' buttons in the top right. The main content area is titled 'Summary' and contains a table with the following information:

Creation date September 23, 2023, 11:22 (UTC-05:00)	ARN <a href="#">arn:aws:iam::300013369769:role/LabRole</a>	Link to switch roles in console <a href="https://signin.aws.amazon.com/switchrole?roleName=LabRole&amp;account=300013369769">https://signin.aws.amazon.com/switchrole?roleName=LabRole&amp;account=300013369769</a>	Instance profile ARN <a href="#">arn:aws:iam::300013369769:instance-profile/LabInstanceProfile</a>
Last activity Yesterday	Maximum session duration 1 hour		

Below the summary table, there are tabs for 'Permissions', 'Trust relationships', 'Tags (1)', 'Access Advisor', and 'Revoke sessions'.

Creamos un schema con el nombre de trabajo1\_schema donde podremos guardar las tablas externas que vayamos creando[código]:

The screenshot shows the 'Redshift query editor v2' interface. On the left, a sidebar shows the resource tree with 'redshift-cluster-1' expanded, and 'trabajo1\_schema' highlighted under the 'public' database. The main editor area shows a SQL query:

```
1 create external schema trabajo1_schema
2 from data catalog
3 database 'trabajo1_db'
4 iam_role 'arn:aws:iam::300013369769:role/LabRole'
5 create external database if not exists;
```

The query is being executed in a session named 'dev' on a cluster named 'redshift-clust...'.

Para poder crear nuestra tabla con los datos que tenemos en nuestra zona trusted, es necesario correr el siguiente código que veremos a continuación donde se debe de pasar el schema de nuestra tabla, así como la ubicación de esta en S3:

Redshift query editor v2

Filter resources

- redshift-cluster-1
  - awsdatacatalog
  - dev
    - public
    - trabajo1\_schema
      - Tables
        - temperaturas\_id...

temperaturas\_ideam

Field	Type	NL
# codigoeleccion	bigint	NULL
# codigosensor	bigint	NULL
A fechaobservacion	varchar(50)	NULL
# valorobservado	double	NULL

```
1 create external table trabajo1_schema.temperaturas_ideam(  
2   codigoeleccion BIGINT,  
3   codigosensor BIGINT,  
4   fechaobservacion varchar(50),  
5   valorobservado DOUBLE PRECISION,  
6   nombreeleccion varchar(50),  
7   departamento varchar(50),  
8   municipio varchar(50),  
9   zonahidrografica varchar(50),  
10  latitud DOUBLE PRECISION,  
11  longitud DOUBLE PRECISION,  
12  descripcionsensor varchar(50),  
13  unidadmedida varchar(50))  
14 STORED AS PARQUET  
15 LOCATION 's3://psariaszlab1/trusted/';  
16
```

Result 1

Summary

Returned rows: 0  
Elapsed time: 513ms  
Result set query:

```
create external table trabajo1_schema.temperaturas_ideam(  
  codigoeleccion BIGINT,  
  codigosensor BIGINT,  
  fechaobservacion varchar(50),
```

Como podemos observar nuestra tabla fue correctamente creada en Redshift con nuestros datos provenientes de S3 en un archivo con formato parquet.

Run Limit 100 Explain Isolated session redshift-clust... dev

```
1 SELECT * FROM trabajo1_schema.temperaturas_ideam;
```

Result 1 (100)

codigoeleccion	codigosensor	fechaobservacion	valorobservado	nombreeleccion	departamento	municipio
23125170	68	2006-09-21T12:00:00.000	15.9	SAN CAYETANO - AUT	CUNDINAMARCA	SAN CAYETANO
16025501	68	2018-12-02T09:20:00.000	23.02354	CUCUTILLA - AUT	NORTE DE SANTANDER	CUCUTILLA
52055501	68	2016-06-25T04:05:00.000	16.3	OSPINA PEREZ - AUT	NARIÑO	CONSACÁ
52055210	68	2014-06-02T08:00:00.000	13.3	BOTANA - AUT	NARIÑO	PASTO
26155502	68	2019-07-27T00:05:00.000	20.45158	CENICAFE - AUT	CALDAS	MANIZALES
21208480	68	2012-03-16T15:15:00.000	19.1	KENNEDY - FOPAE	BOGOTA D.C.	BOGOTA, D.C.
21208480	68	2016-01-17T23:55:00.000	16.6	KENNEDY - FOPAE	BOGOTA D.C.	BOGOTA, D.C.
24035390	68	2015-02-24T00:00:00.000	2.9	PARAMO ALMORZADER...	SANTANDER	CERRITO
21205519	68	2019-04-20T03:50:00.000	11.072	SAN FORTUNATO - AUT	CUNDINAMARCA	SIBATÉ
23055501	68	2017-10-10T22:50:00.000	19.13098	SANTA HELENA - AUT	CALDAS	MARQUETALIA

Realicemos una consulta SQL sencilla dentro de Redshift para probar el funcionamiento de esta y como nos puede brindar datos de análisis, como lo hicimos con Athena, Hive o PySpark:

+
Untitled 1 x
Untitled 2 x
Untitled 3 x

Run
Limit 100
Explain
Isolated session
redshift-clust...

```

1 SELECT
2     nombreestacion,
3     AVG(valorobservado) AS average_valorobservado, departamento
4 FROM
5     trabajo1_schema.temperaturas_ideam
6 GROUP BY
7     nombreestacion, departamento
8 ORDER BY
9     average_valorobservado DESC;

```

Result 1 (100)

	nombreestacion	average_valorobservado	departamento
<input type="checkbox"/>	HACIENDA PAJONALES	35.7	TOLIMA
<input type="checkbox"/>	EL DIFICIL - AUT	34.25	MAGDALENA
<input type="checkbox"/>	LOS ALAMOS	33.4	MAGDALENA
<input type="checkbox"/>	FEDEARROZ - AUT	33.349999999999994	CESAR
<input type="checkbox"/>	APTO ERNESTO CORTI...	31.7	ATLÁNTICO
<input type="checkbox"/>	APTO ALMIRANTE PADI...	30.8	LA GUAJIRA
<input type="checkbox"/>	PAICI - AUT	30.566666666666663	LA GUAJIRA
<input type="checkbox"/>	ESTRECHO PATIA - AUT	30.45	CAUCA
<input type="checkbox"/>	APTO ALFONSO LOPEZ	30.4	CESAR
<input type="checkbox"/>	REPELON - AUT	30.333333333333332	ATLANTICO

Ahora para llegar al punto que nos interesa en Redshift, y es el de un Datawarehouse Moderno, donde adicional al almacenamiento y procesamiento de nuestros datos también podemos crear mediante sentencias SQL un modelado de entrenamiento; El modelado de entrenamiento para este ejercicio se basara en que, dependiendo del Departamento, Redshift me pueda brindar un valor observado, es decir una temperatura objetivo que esta podría tener en el tiempo. [\[código\]](#)

```
1 CREATE MODEL temp_observado_auto_model_v2 FROM (SELECT codigoestacion,
2 FECHA,
3 valorobservado,
4 departamento,
5 latitud,
6 longitud
7 FROM public.temp_definitiva
8 WHERE FECHA < '2018-01-01')
9 TARGET valorobservado FUNCTION ml_fn_temp_observado_v2
10 IAM_ROLE 'arn:aws:iam::300013369769:role/LabRole'
11 SETTINGS (
12 | S3_BUCKET 'psariaszlab1'
13 );
```

**Result 1**

### Summary

Returned rows: 0  
Elapsed time: 1.7s  
Result set query:

```
CREATE MODEL temp_observado_auto_model_v2 FROM (SELECT codigoestacion,
FECHA,
valorobservado,
departamento,
latitud,
longitud
FROM public.temp_definitiva
WHERE FECHA < '2018-01-01')
```

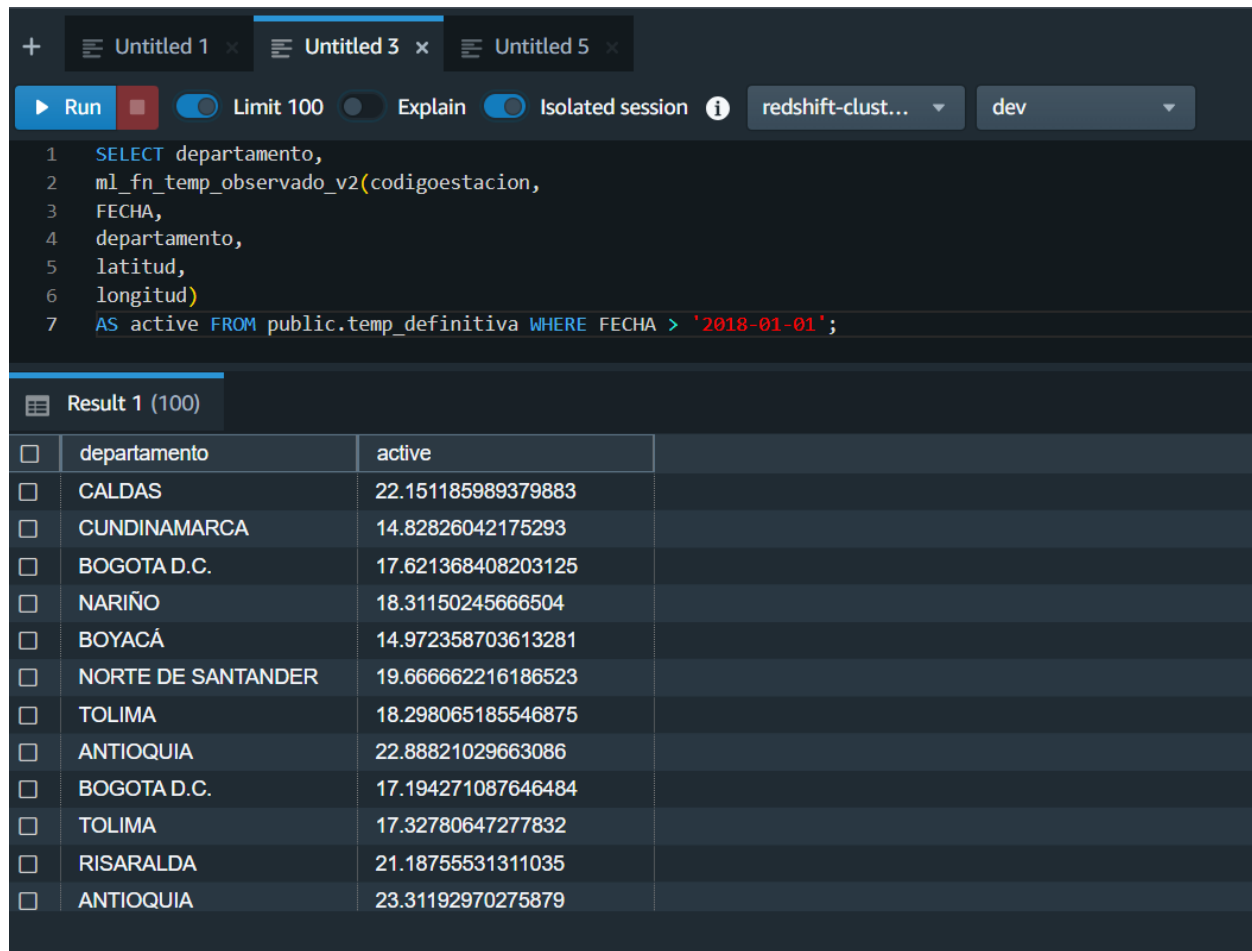
A continuación, podemos ver, como con la siguiente consulta de SQL podemos conocer el estado de mi modelo, que para ese momento se encontraba en entrenamiento:

```
1 SELECT * FROM STV_ML_MODEL_INFO;
```

**Result 1 (5)**

_name	model_name	life_cycle	is_refreshable	model_state	database
user	temp_observadas_model ...	Active	1	Model Failed	dev
user	temp_observado_auto_m...	Active	1	Train Model In Progress	dev
user	temperaturas_ideam	Active	1	Model Failed	dev
user	temp_observadas_model...	Active	1	Model Is Ready	dev
user	temp_observado_auto_m...	Active	1	Train Model In Progress	dev

Por último, aplicamos nuestro modelo para observar que posibles valor observado pueden llegar a tener cada departamento a partir del 2018. En este caso por fines prácticos seleccionamos solo la variable departamento, pero también puede ser mezclado con variables como municipios para obtener información mas precisa.



The screenshot shows a Redshift SQL query editor with three tabs: 'Untitled 1', 'Untitled 3', and 'Untitled 5'. The 'Untitled 3' tab is active. The query is as follows:

```
1 SELECT departamento,
2 ml_fn_temp_observado_v2(codigoestacion,
3 FECHA,
4 departamento,
5 latitud,
6 longitud)
7 AS active FROM public.temp_definitiva WHERE FECHA > '2018-01-01';
```

Below the query, the 'Result 1 (100)' tab is selected, displaying a table with two columns: 'departamento' and 'active'. The table contains 12 rows of data, with the first row being the header and the subsequent rows showing department names and their corresponding active values.

departamento	active
CALDAS	22.151185989379883
CUNDINAMARCA	14.82826042175293
BOGOTA D.C.	17.621368408203125
NARIÑO	18.31150245666504
BOYACÁ	14.972358703613281
NORTE DE SANTANDER	19.666662216186523
TOLIMA	18.298065185546875
ANTIOQUIA	22.88821029663086
BOGOTA D.C.	17.194271087646484
TOLIMA	17.32780647277832
RISARALDA	21.18755531311035
ANTIOQUIA	23.31192970275879