Data Engineering Flex

Profesor: Augusto Castro

Tutor: Leandro Eguía

Alumno: Rafael Wierna

Comisión: 55160

Tema obtención de información sobre clima de varias ciudades de Argentina



2023

**Objetivo**

Obtención de información sobre clima de varias ciudades de Argentina, donde se cree un pipeline que extraiga datos de una API pública de forma constante

Colocar los datos extraídos en un Data Warehouse en la nube.

Automatizar el proceso que extraerá, transformará y cargará datos cuantitativos

Automatizar el proceso para lanzar alertas por e-mail enviando el status del ETL

**Requerimientos**

1. Docker
2. Airflow
3. REDSHIFT
4. API sobre clima
5. Mail

**Propósito**

Realizar un ETL que

1. Solicite datos del clima de un conjunto de ciudades de Argentina diariamente
   1. Anexo 2
2. Que realice la solicitud de los últimos 7 días (restricción de la api seleccionada)
   1. fechas = [(fecha\_hoy - timedelta(days=i)).strftime('%Y-%m-%d') for i in range(7)]
3. Que la solicitud lo haga mediante una API
   1. https://weatherapi-com.p.rapidapi.com/history.json
4. Que realice backups diarios previa a la corrida
   1. Tabla pronostico\_clima\_backup: se agrega fecha de backup para realizar controles
      1. Fecha\_backup: fecha en la que se realiza el backcup
5. Que en la solicitud no procese datos de ciudades que ya se procesaron en esa fecha
   1. Posee una fecha de refencia de backups, el campo fecha\_alta\_proceso
6. Almacenar los datos obtenidos diariamente en la nube
   1. data-engineer-cluster.cyhh5bfevlmn.us-east-1.redshift.amazonaws.com
   2. Tabla pronostico\_clima: posee dos campos con fechas
      1. Fecha\_alta\_proceso: fecha en la que se cargó el registro
      2. Fecha: fecha a la cual pertenece el dato solicitado
7. Monitorear la ejecución de procesos
8. Enviar mails de alarmas sobre el estado del proceso
9. Almacenar logs sobre lo sucedido en los diferentes procedimientos ejecutados
   1. Tabla pronostico\_logs
      1. fecha\_alta\_proceso : fecha de alta del registro
      2. id\_msj: posee la fecha y ahora para saber el orden en lo que sucedieron las ejecuciones o ingresos de datos
      3. msj: texto explicativo de lo que sucedió
10. Mails con el resultado sobre el status final del procedimiento, se agrega fecha y hora en el cuerpo del correo Anexo 5
11. Análisis de datos con panda, sobre el archivo de BD que posee los pronósticos Anexo 6

**Descripción**

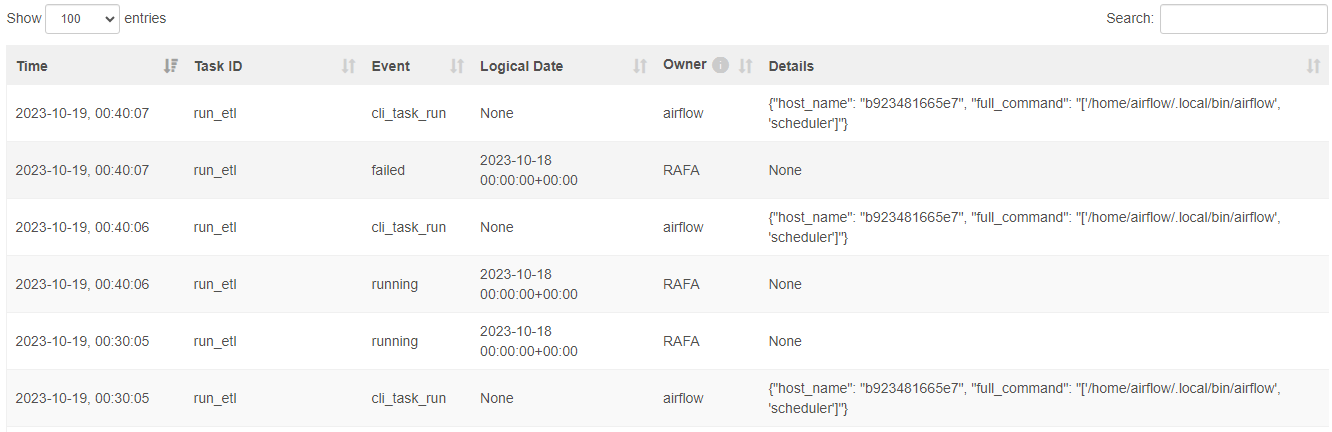
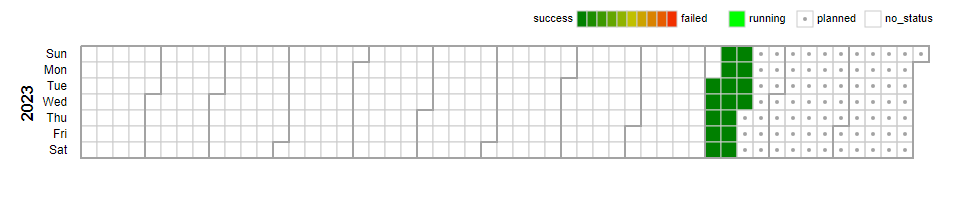
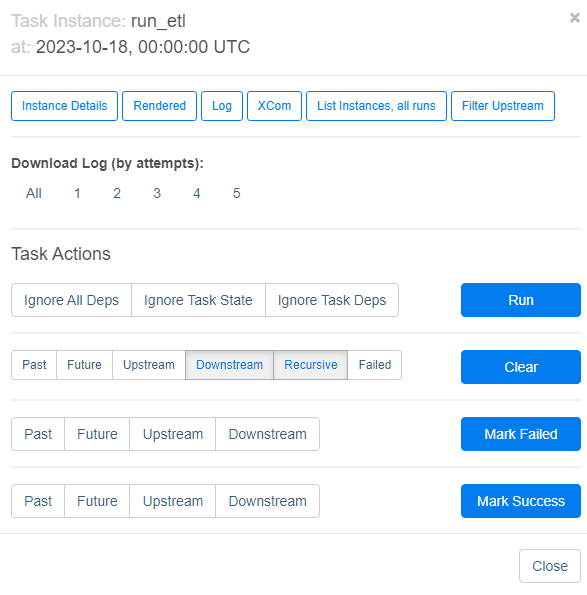
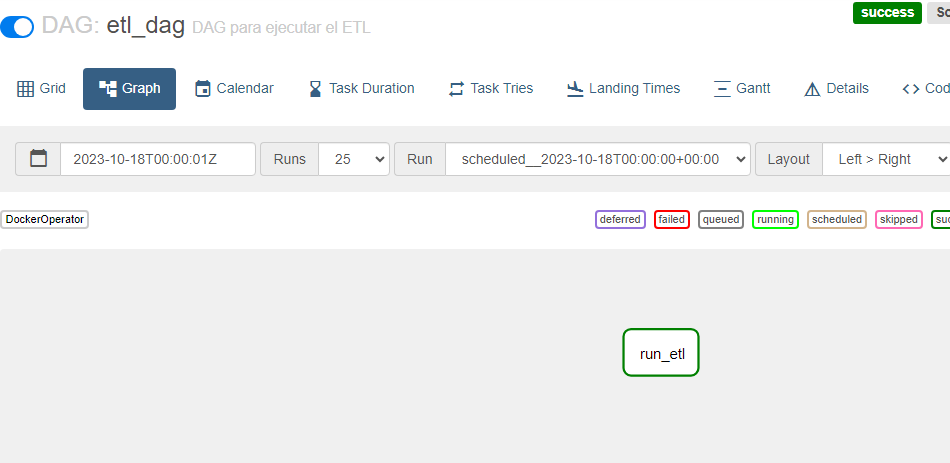
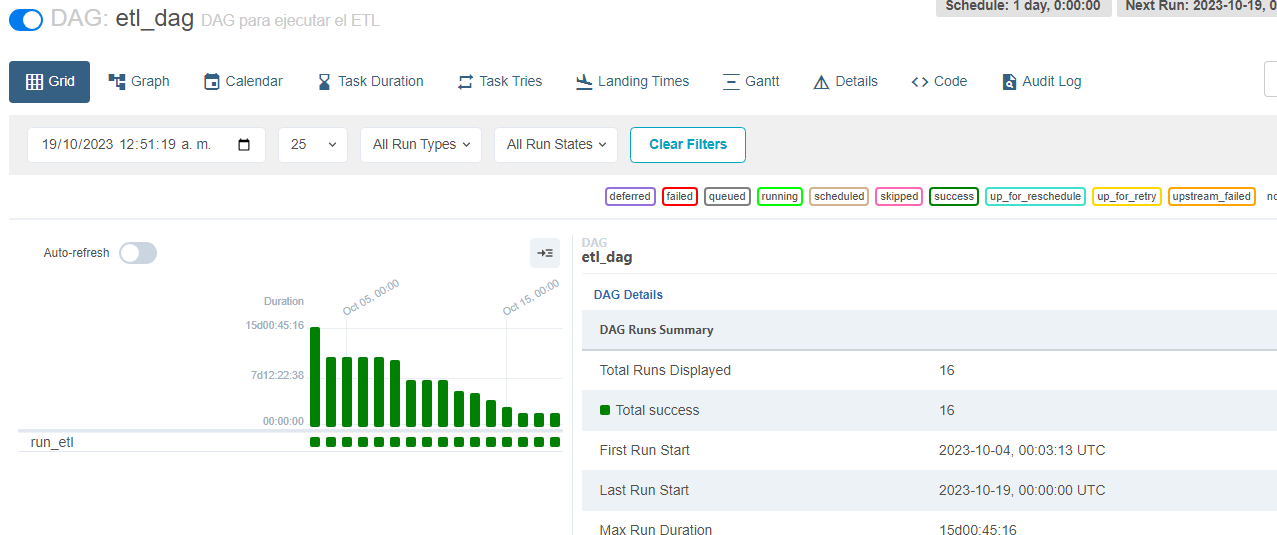
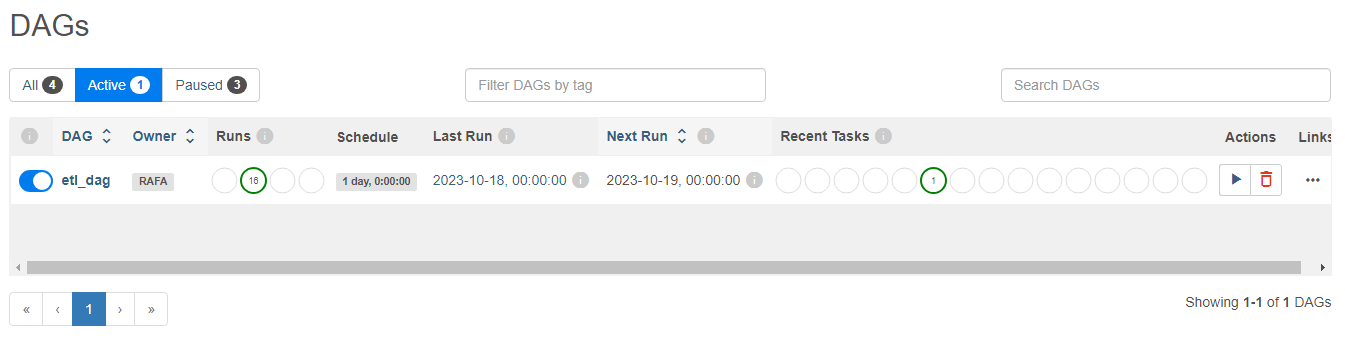
El código del ETL, que se llama 04-Cuarta entrega.py, posee la siguiente descripción

1. Importación de librerías necesarias requests, psycopg2, airflow, smtplib , etc..
2. Toma la hora de inicio del programa con la intención de saber el costo en tiempo del procesamiento
3. Inicializamos variables
4. Configuramos Redshift
5. Realizamos la conexión
6. Configuramos la API del tiempo elegida
7. Configuramos los datos del servidor SMTP para enviar mail
8. Configuramos el mensaje
9. Definimos la estructura de la tabla donde almacenaremos la info obtenida ANEXO 1
10. Realizamos el backup de los datos históricos ya almacenado para insertar los nuevos en pronostico\_clima\_backup, simpre que no se haya realizado una de esa fecha ANEXO 4
11. Obtenemos el máximo ID de los registros para saber desde cual continuar
12. Definimos el listado de ciudades de las cuales vamos a averiguar sus datos del clima Anexo 2
13. Definimos las Fechas para las que se desea obtener el pronóstico, este api posee la restricción de 7 días solamente de antigüedad
14. Itera a través de las ciudades y las fechas
15. Luego comienza la solicitud de datos siempre que no se encuentren ya cargada la fecha y la ciudad
16. Comprueba si la solicitud fue exitosa (código de estado 200)
17. Almacenamos los datos del pronóstico en el diccionario cities\_weather\_data
18. Insertamos en pronostico\_clima
19. En caso de error envía mail de alerta
20. Luego de iterar por las 22 ciudades finaliza con un mail de procesos satisfactorio
21. También guarda en otra tabla de logs (Pronostico\_logs) lo que va sucediendo mientras se ejecuta el proceso ETL (Anexo 3)

**DAG**

El archivo se llama ETL\_DAG , contiene una sola tarea

La cual invoca al proceso de ETL



**from** **airflow** **import** DAG

**from** **datetime** **import** datetime, timedelta

**from** **airflow.operators.docker\_operator** **import** DockerOperator

default\_args = {

'owner': 'RAFA',

'depends\_on\_past': **False**,

'start\_date': datetime(2023, 10, 3),

'email': 'nrwnet@gmail.com',

'email\_on\_failure': **False**,

'email\_on\_retry': **False**,

'retries': 5,

'retry\_delay': timedelta(minutes=10),

}

dag = DAG(

'etl\_dag',

default\_args=default\_args,

description='DAG para ejecutar el ETL',

schedule\_interval=timedelta(days=1),

)

run\_etl = DockerOperator(

task\_id='run\_etl',

image='apache/airflow', *# Nombre de la imagen Docker*

command='python 03-Tercera entrega.py', *# Comando para ejecutar el script dentro del contenedor*

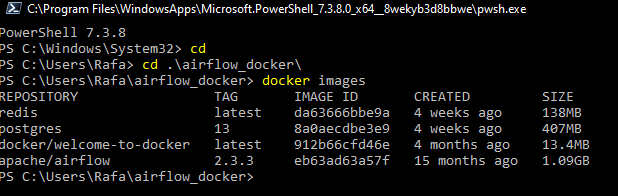
api\_version='auto',

auto\_remove=**True**,

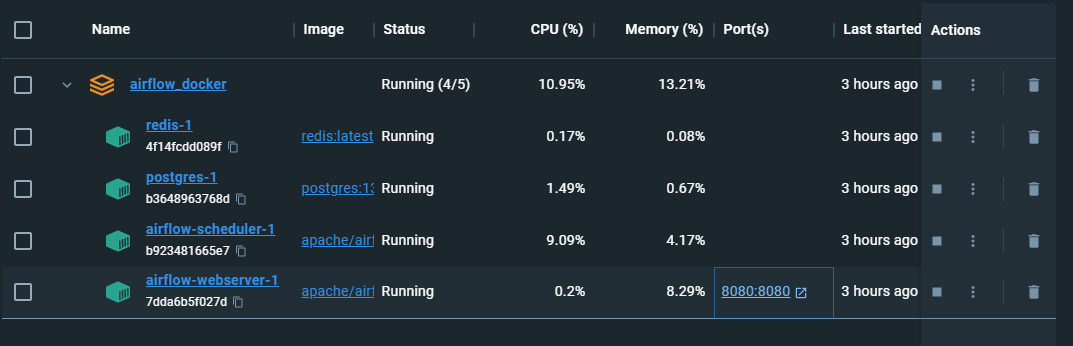
dag=dag,

)

**Captura de imágenes**



**Docker**



**Anexo 1**

Estructura de tabla donde se almacenas los datos del clima de las ciudades

pronostico\_clima(

id\_registro varchar,

ciudad varchar,

region varchar,

pais varchar,

latitud varchar,

longitud varchar,

descripcion\_clima VARCHAR,

humedad VARCHAR,

fecha VARCHAR,

temperatura\_minima VARCHAR,

temperatura\_maxima VARCHAR,

velocidad\_viento VARCHAR,

direccion\_viento VARCHAR,

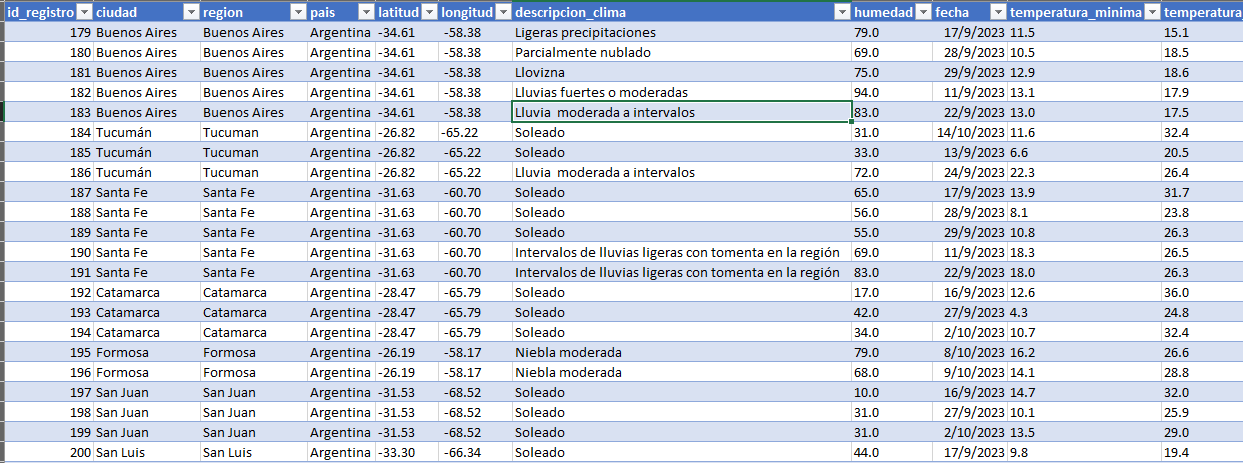
presion\_atmosferica VARCHAR,

indice\_uv VARCHAR,

visibilidad VARCHAR,

precipitacion VARCHAR,

fecha\_alta\_proceso varchar



**Anexo 2**

Datos de las ciudades que se solicitara información del clima

#Diccionario de datos de ciudades

cities\_data = {

'Buenos Aires': '-34.61, -58.38',

'Catamarca': '-28.47, -65.79',

'Córdoba': '-31.41, -64.18',

'Mendoza': '-32.89, -68.84',

'Rosario': '-32.95, -60.65',

'Salta': '-24.79, -65.41',

'Tucumán': '-26.82, -65.22',

'La Plata': '-34.92, -57.95',

'Santa Fe': '-31.63, -60.70',

'San Juan': '-31.53, -68.52',

'San Luis': '-33.30, -66.34',

'Neuquén': '-38.95, -68.05',

'Bahía Blanca': '-38.71, -62.27',

'Mar del Plata': '-38.00, -57.56',

'Tandil': '-37.32, -59.14',

'Misiones': '-27.37, -55.90',

'Chaco': '-27.46, -58.98',

'Formosa': '-26.19, -58.17',

'Jujuy': '-24.19, -65.30',

'Corrientes': '-27.47, -58.84',

'Santa Cruz': '-51.62, -69.22',

'Tierra del Fuego': '-54.80, -68.30'

}

**Anexo 3** – Estructura de la tabla LOGS

pronostico\_logs(

id\_log varchar,

fecha\_alta\_proceso VARCHAR,

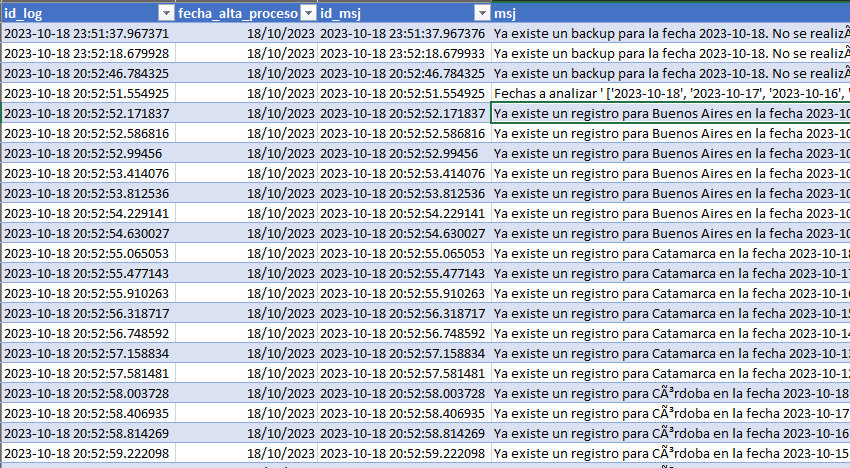
id\_msj VARCHAR,

msj VARCHAR,

);

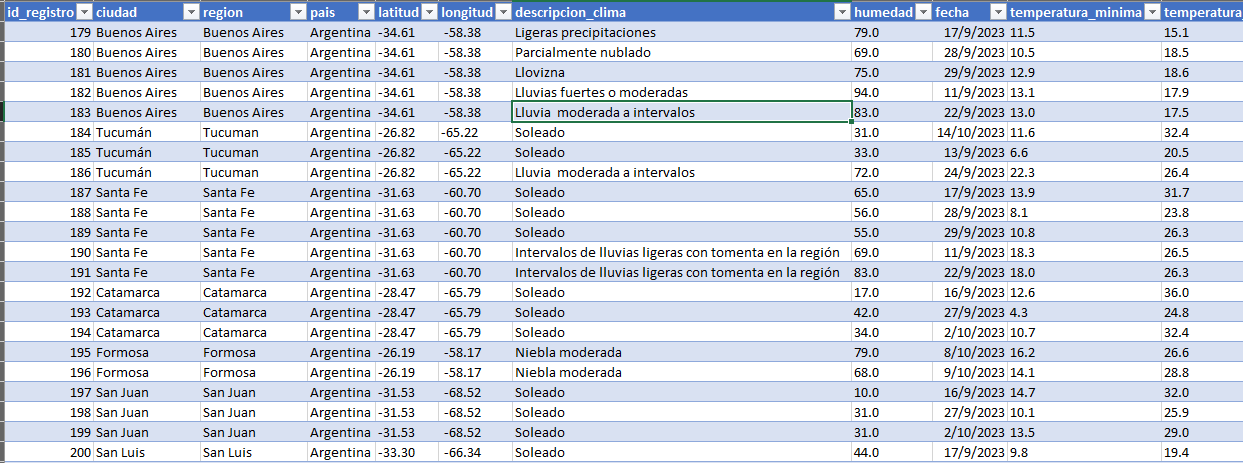
"""

**Muestreo de Información almacenada**



Anexo 4 estructura tabla Backup

Es la misma que la tabla principal, donde se le agrega la fecha del dia del backup



Anexo 5 – mail

texto = f"""Inicio Status ETL,

    Finalizo correctamente

    Fecha y hora {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

    Fin Status ETL

    """

Anexo 6

#analisis de datos

# Consulta SQL para obtener los datos

sql\_query = "SELECT \* FROM pronostico\_clima;"

#lectura

df = pd.read\_sql\_query(sql\_query, conn)

type(df)

df.shape

df.head()

df.tail()

df.info()

print (df.isnull().sum())

print (df.notnull().sum())

df.isna().sum()

df.describe(include='all')

df.describe().T

Resultado

RangeIndex: 934 entries, 0 to 933

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id\_registro 934 non-null object

1 ciudad 934 non-null object

2 region 934 non-null object

3 pais 934 non-null object

4 latitud 934 non-null object

5 longitud 934 non-null object

6 descripcion\_clima 934 non-null object

7 humedad 934 non-null object

8 fecha 934 non-null object

9 temperatura\_minima 934 non-null object

10 temperatura\_maxima 934 non-null object

11 velocidad\_viento 934 non-null object

12 direccion\_viento 930 non-null object

13 presion\_atmosferica 934 non-null object

14 indice\_uv 934 non-null object

15 visibilidad 934 non-null object

16 precipitacion 934 non-null object

17 fecha\_alta\_proceso 933 non-null object

dtypes: object(18)

memory usage: 131.5+ KB

id\_registro 0

ciudad 0

region 0

pais 0

latitud 0

longitud 0

descripcion\_clima 0

humedad 0

fecha 0

temperatura\_minima 0

temperatura\_maxima 0

velocidad\_viento 0

direccion\_viento 4

presion\_atmosferica 0

indice\_uv 0

visibilidad 0

precipitacion 0

fecha\_alta\_proceso 1

dtype: int64

id\_registro 934

ciudad 934

region 934

pais 934

latitud 934

longitud 934

descripcion\_clima 934

humedad 934

fecha 934

temperatura\_minima 934

temperatura\_maxima 934

velocidad\_viento 934

direccion\_viento 930

presion\_atmosferica 934

indice\_uv 934

visibilidad 934

precipitacion 934

fecha\_alta\_proceso 933

dtype: int64