En esta clase conoceremos cómo cargar datos en formato json a Redshift con el comando Copy, ya conoces las ventajas de este comando, ahora veremos cómo llevar nuestros datos a Redshift dado el caso que nuestro origen de datos se encuentre en un formato json.

Lo primero que debemos saber es que existen dos formas de cargar los datos en formato json en Redshift, una es la forma automática, en donde cada "key" del archivo json debe hacer match con cada columna que queramos llenar de una tabla en Redshift.

Es importante saber que las columnas en Redshift se crean en minúsculas, de manera que las "key" del archivo en json deben estar en minúscula también. La segunda forma es especificando la estructura del archivo para que este pueda ser cargado a la tabla, para ello utilizaremos un archivo extra denominado jsonpaths.

Carga automática:

Usando la estructura de tabla que hemos manejado en este curso:

```
--Create table

create  table  estudiante

( id  int2,

nombre  varchar(20),

apellido  varchar(20),

edad  int2,

fecha_ingreso  date );
```

Crearemos una estructura en Redshift que pueda satisfacer las necesidades de esta tabla, de modo que puede quedar algo así (recuerda que los "key" del archivo se debe llamar igual que en la tabla):

```
{
        "id": 4544,
        "nombre": "test_1",
        "apellido": "json_1",
        "edad": 33,
        "fecha_ingreso": "2020-08-01"
}
{
        "id": 23232,
        "nombre": "test_2",
        "apellido": "json_2",
        "edad": 22,
```

```
        "fecha_ingreso": "2020-08-03"
}
```


Ahora lo subimos a S3 con el nombre que queramos y procedemos a cargarlo de la siguiente manera:

copy public.estudiante from 's3://[tu_propia_ruta_del_archivo_json]' credentials 'aws_iam_role=[tu_iam_role]'

format as json 'auto' region '[tu_region]';

Como se ve en la gráfica, la tabla se encuentra cargada con los datos que existen en el archivo json.

Carga con jsonpaths:

Esta carga consiste en determinar una estructura basada en el archivo json que pueda ser insertada en la tabla, como sabemos en un archivo json podemos tener datos no estructurados, pero Redshift requiere de data estructurada; de manera que lo primero será crear un pequeño ejemplo de un archivo json con datos no estructurada.


```
{
        "id": 4544,
        "nombre": "test_json3",
        "apellido": "test_json3",
        "edad": [
                24332,
                33,
                443,
                323232,
                43434
        ],
        "fechas": [
                {
                        "id": 0,
                        "fecha_ingreso": "2015-05-01"
                },
                {
                        "id": 1,
```

```
                "fecha_ingreso": "2016-05-30"
            }
        ]
}
```

Vamos a suponer que de la lista de edades son la segunda posición o sea la posición [1] es la que yo requiero, y para las fechas, la fecha que en verdad me interesa es la contenida en el segundo bloque, para especificar estas reglas crearemos un nuevo archivo con la siguiente estructura y lo nombraremos “jsonpaths”:

```
{
    "jsonpaths": {
        "$['id']",
        "$['nombre']",
        "$['apellido']",
        "$['edad'][1]",
        "$['fechas'][1]['fecha_ingreso']"
    }
}
```

En donde el símbolo ($) indica la raíz del archivo json.

Ahora subimos nuestros dos archivos a S3 y ejecutamos la siguiente sentencia en nuestro editor SQL.

copy public.estudiante from 's3://[tu_propia_ruta_del_archivo_json]' credentials 'aws_iam_role=[tu_iam_role]'

format as json 's3://[tu_propia_ruta_del_archivo_jsonpaths]' region 'tu_propia_region';

Y como vemos, el registro fue cargado exitosamente.

id	nombre	apellido	edad	fecha_ingreso
4.544	test_1	json_1	33	2020-08-01
23.232	test_2	json_2	22	2020-08-03
4.544	test_json3	test_json3	33	2016-05-30