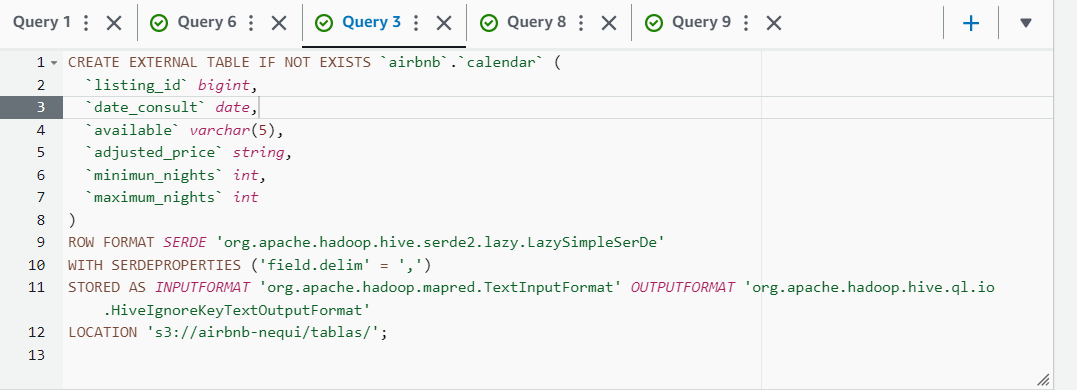
Paso 1: Alcance del proyecto y captura de datos.

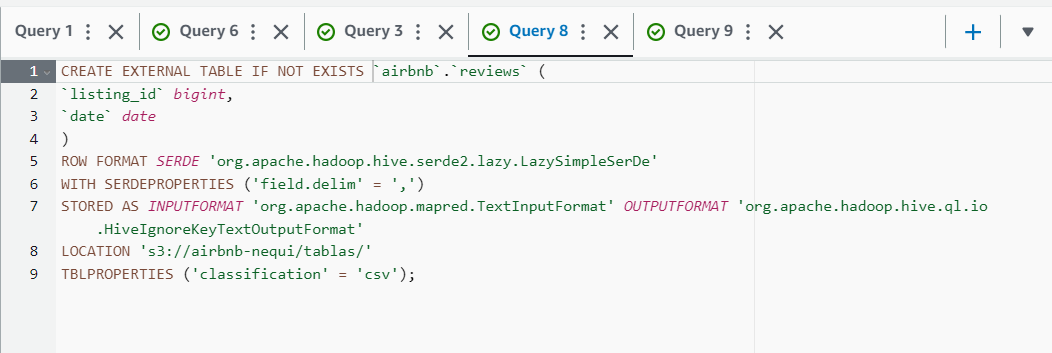
1. Selección de dataset: Se selecciono un dataset que contiene información de las propiedades disponibles para renta en la ciudad de Nueva york, se tienen 3 tablas disponibles, la primera contiene toda la información de los review de cada una de las propiedades en la pagina web, esta se llama Reviews; la segunda tabla tiene toda la información de disponibilidad de las propiedades, esta tabla se llama Calendar; la tercera tabla contiene información general de cada propiedad como propietario, ubicación, precio por noche, descripción ect, esta tabla se llama Listings.
2. Los casos de uso finales para la preparación de estos datos son los siguientes: el objetivo principal es crear un panel de control (dashboard) que proporcione información clara y concisa sobre la cantidad de visitas de propiedades en Airbnb en la ciudad de Nueva York. Además, este panel de control permitirá acceder a las especificaciones detalladas de cada propiedad, lo que brindará un control inmediato sobre la información y facilitará la creación de estrategias para potenciar diferentes sectores de la ciudad en términos de arrendamiento.

Con este dashboard, los usuarios podrán obtener una visión general de la cantidad de visitas de las propiedades en Airbnb, lo cual les permitirá evaluar rápidamente la popularidad y demanda de estas. Asimismo, podrán acceder a las especificaciones detalladas de cada propiedad, como la ubicación, el tamaño, las comodidades y otros detalles relevantes. Esto les dará una imagen completa de las características de las propiedades y les permitirá tomar decisiones informadas en cuanto a estrategias para potenciar diferentes sectores de la ciudad en relación con el arrendamiento.

Paso 2: Explorar y evaluar los datos, el EDA.

Para realizar la exploración de los datos, se realizo una carga de los archivos en formato csv a un bucket de S3, esto se hizo a través de un código sencillo en Python que usa la librería *boto3*, las tablas luego de estar en el bucket llamado Airbnb-nequi se procedió a llevarlas el servicio de Athena, a través de unos códigos en SQL HIVE.



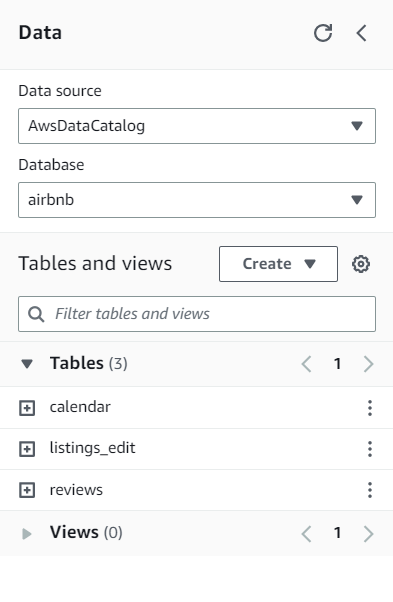


Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Una vez con las tablas cargadas en Athene en la base de datos cruda llamada: airbnb

, procedemos a hacer los análisis correspondientes a la calidad de los datos.



Una vez las tablas estén cargadas en athena, se procede a realizar validaciones de calidad de datos.

Ejemplos de querys de validación de calidad de datos:

Validacion de nulidad en llaves

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

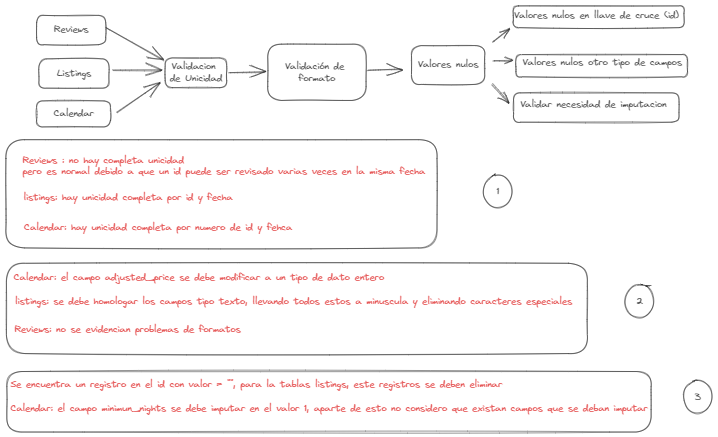
Validacion de unicidad:

Imagen que contiene Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



Paso 3: Definir el modelo de datos

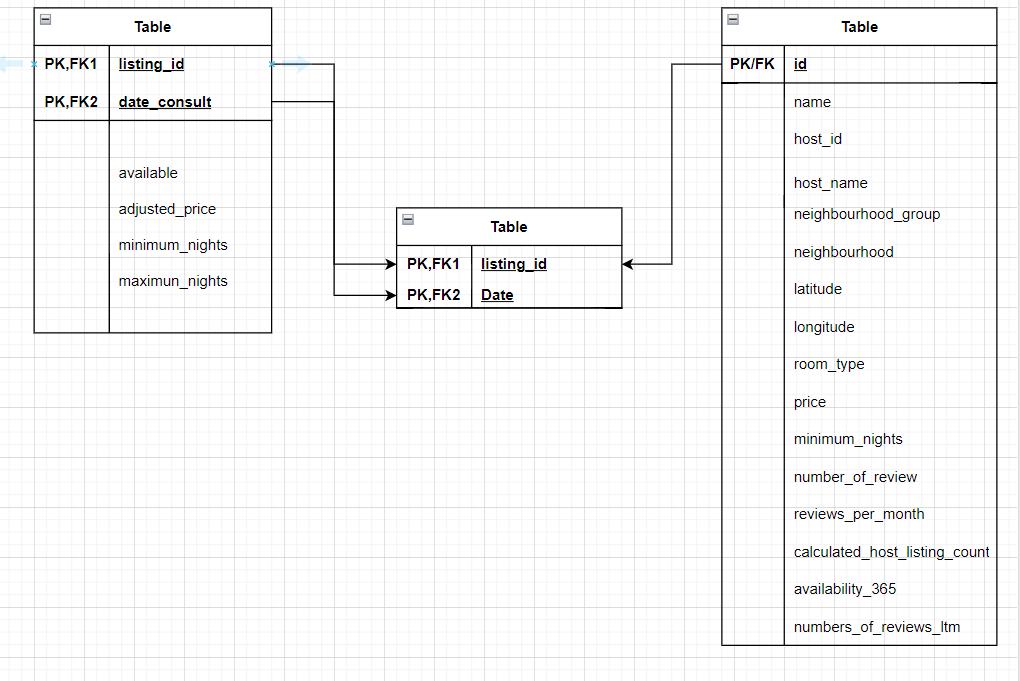
Se define un modelo de datos dimensional tipo estrella, donde nuestra tabla de hecho va a ser la tabla de reviews, se escoge este modelo por las siguientes razones:

* Es un modelo muy usado en la construcción de data warehousing.
* Simplicidad y comprensión gracias a su estructura centralizada a una tabla de hecho.
* Rendimiento y consulta eficiente debido a su diseño simplificado.
* Flexible y escalable, es relativamente sencillo agregar otra dimensión al modelo sin afectar las demás dimensiones.
* Análisis y reporteria: El modelo estrella se adapta muy bien a las necesidades de análisis y reporteria ya que los datos se organizan de manera sencilla para realizar consultas sobre ellas.

Arquitectura:

Diagrama, Esquemático

Descripción generada automáticamente



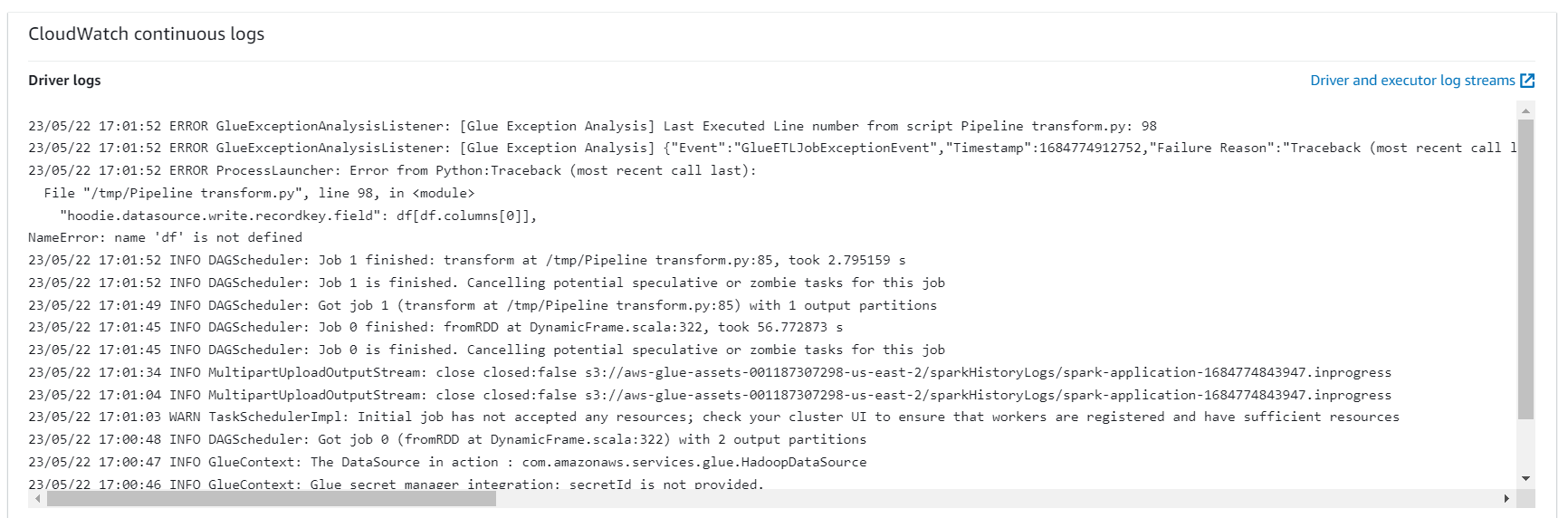
Para el desarrollo del proyecto se instanciaron los siguientes servicios:

AWS S3 storege: Este lo usamos como servicio de almacenamiento de nuestro dataset, S3 te permite organizar tus datos en "buckets" (cubetas), lo que facilita la gestión y el acceso a los archivos además de permitirnos manejar un gran volumen de datos.

AWS Athena: Este servicio lo usamos para realizar un reconocimiento de nuestro datalake, desde acá observaremos las transformaciones que se deben hacer a nuestros datos. Athena es un servicio que nos permite conectarnos muy fácilmente a nuestros archivos que se encuentran en nuestros buckets de S3 sin necesidad de gestionar nuestras bases de datos, también es un servicio altamente escalable que puede procesar consultas de manera distribuida y paralela.

AWS Glue: Este servicio lo usamos para desarrollar todas nuestras trasformaciones sobre nuestro set de datos y también para crear nuestros controles de calidad de datos, este permite trabajar con Python o pyspark de forma distribuida, lo que hace más rápido para la ejecución de nuestros pipeline, además que es altamente escalable lo que nos permite trabajar con un gran volumen de datos y nos permite una fácil integración con los demás servicios aws, en este caso con S3.

AWS cloudwatch: nos permite capturar, almacenar y analizar los registros generados por nuestras aplicaciones y recursos de aws. Podemos buscar y filtrar registros, realizar análisis de patrones, identificar problemas de manera eficiente. Adjunto una imagen de los logs generadores en tiempo de ejecución de un error que se obtuvo probando nuestro pipeline.



Cada cuanto se debe actualizar la información de las tablas.

De acuerdo al modelo de datos que se definió, los datos de nuestra tabla de reviews la cual es nuestra tabla de hecho, se debe actualizar diariamente, debido a que los datos se usaran para alimentar un Dashboard para realizar análisis de mercadeo, las otras dos tablas se deben actualizar solo cuando hayan nuevos inmuebles matriculados, o hayan cambios en la data ahora existente en ambas tablas.

Paso 4: ejecutar etl

Se usa el servicio de AWS Glue para crear el pipeline que se encargara de hacer los ajustes detectados anteriormente con la ayuda de Athena, antes de acceder a los servicios debemos crear un ROL que nos permite acceder a todos los recursos del cloud, una vez terminado creamos un job, el cual será primero un notebook, esté nos ayudara a probar nuestro código a medida que se vayan creando los respectivos transformadores para asociarlos al pipeline, el código del pipeline se desarrollo en Pyspark con ayuda de la librería de Pyspark.ml, esta nos proporciona dos clases las cuales usaremos, la primera llamada Pipeline que es la que nos permite llevar el flujo de la ejecución de las diferentes transformaciones, y la segunda llamada Transformer, esta nos proporciona una clase abstracta donde realizamos todos los ajustes necesarios en nuestro codigo.

Una vez terminado de probar desarrollar nuestro código en el notebook de glue, desplegamos un script donde almacenaremos nuestro código fuente, a continuación, una imagen donde se muestra los dos servicios usados, nuestro job ya creado y el éxito de la ejecucion:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

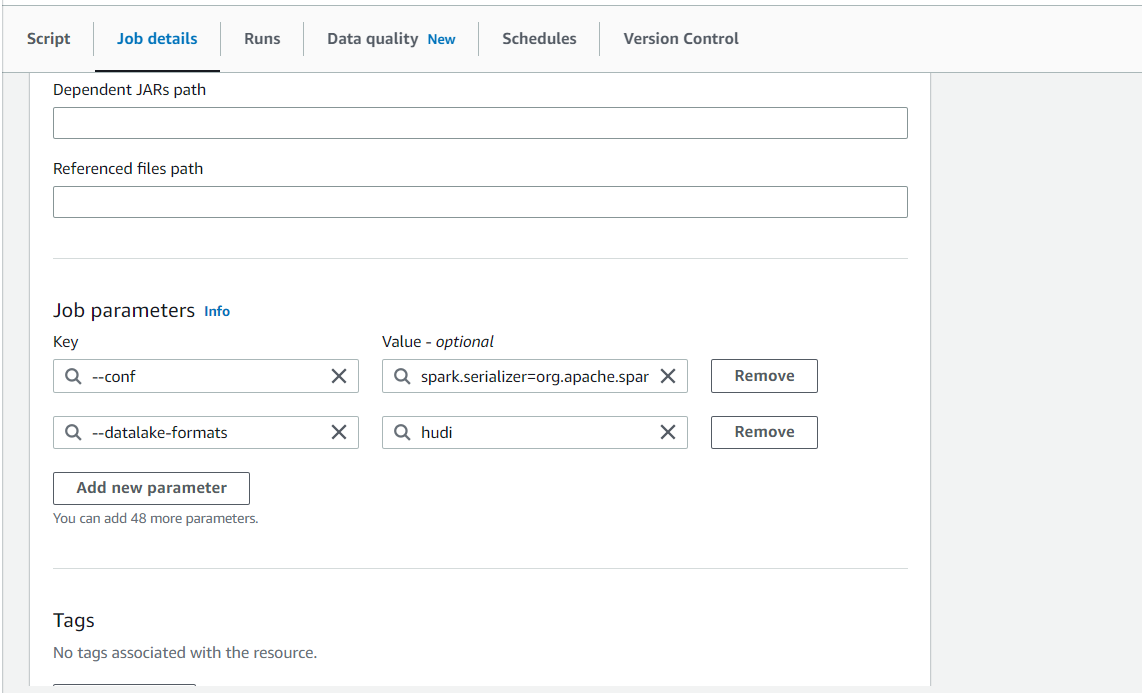
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

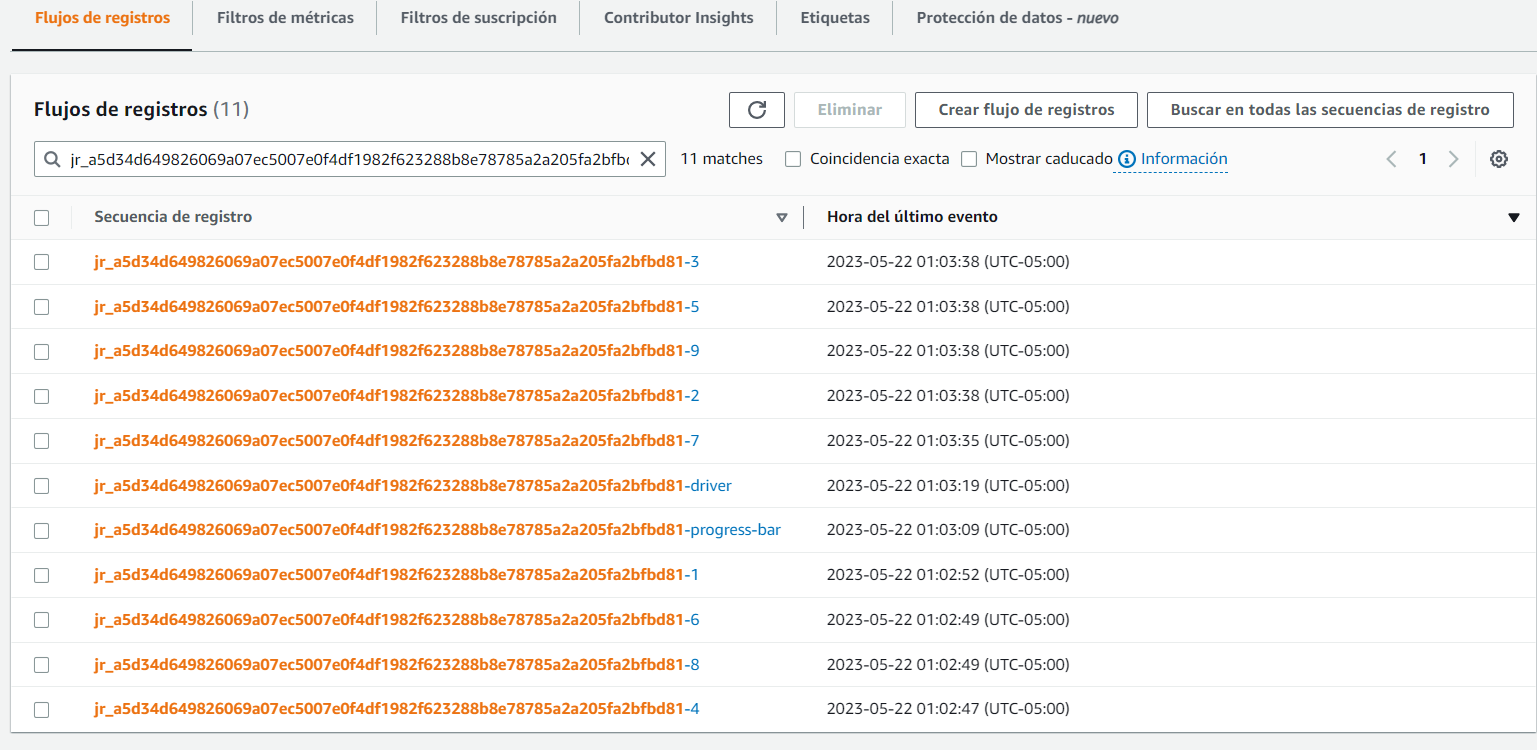
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

El paso final de nuestro Job es crear 3 nuevos archivos ya con todos los ajustes terminados y llevarlos al bucket de S3, esto con la ayuda del framework hudi, se hace la siguiente configuración en los parámetros del job para poder usarlo.

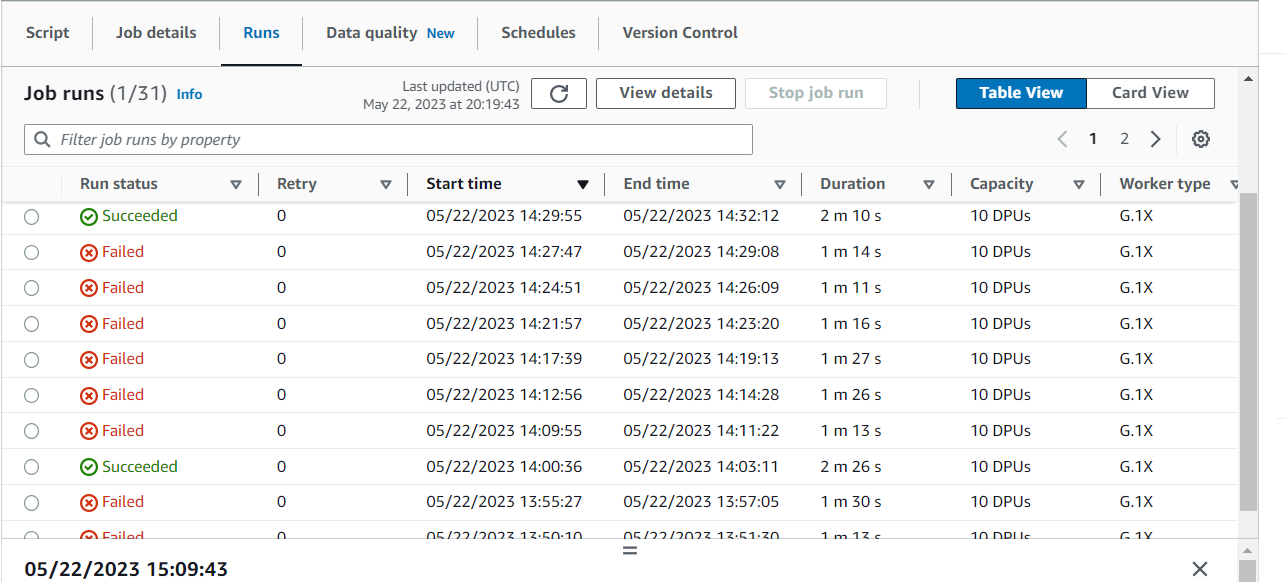


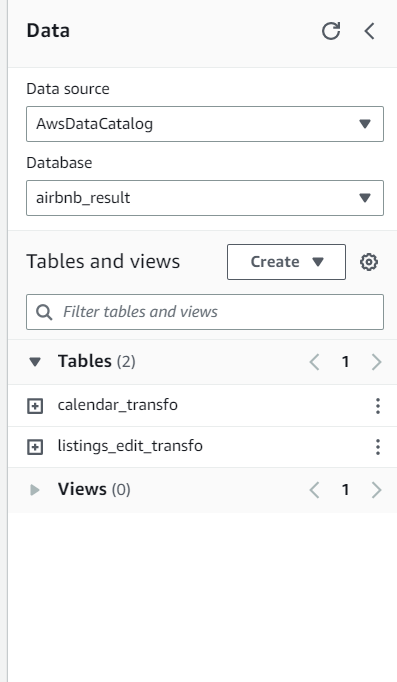
Para el manejo de los logs, se usó el servicio de CloudWatch servicio con el cual podemos llevar un control de ejecución del pipeline y validar que se ejecutó correctamente.



Terminado de ejecutar nuestro pipeline, se evidencia que hizo lo que esperaba, que es construir dos tablas nuevas en Athena en una base de datos específica para los datos tratados que es Airbnb\_result, una para listings\_edit y otra para calendar, nuestra tabla reviews no la hacemos pasar por el pipeline ya que no muestra problemas que se deban corregir.

A continuación, una imagen de que el pipeline ejecuto correctamente después de varios intentos, una imagen donde se evidencia la creación de estos archivos en S3 y una imagen donde se vean las tablas creadas en Athena:





Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Con esto se da por terminado el item 1 del paso 4.

Para validar que la tubería funciono correctamente, se harán unas comparaciones de las tablas anteriores contra las actuales en un job script de glue.

Realizamos la carga de las tablas crudas y las tablas transformadas desde el catálogo de athena.

Ejecutar controles de calidad de los datos para asegurar que la tubería funcionó como se esperaba.

Para este punto se definieron cuatro variables validadoras: validador\_1, validador\_2, validador\_3, validador\_4, estas acceden a distintas funciones que realizan diferentes tareas. La primera valida registros nulos, la segunda valida caracteres especiales, la tercera valida mayúsculas en una lista de columnas definidas y el 4 valida que la columna minimum\_nights efectivamente no tenga valores en blanco. Si alguna de estas variables encuentra algún problema en estos ítems retorna un valor de 1, si funciona correctamente retorna un valor de 0. Al final de la ejecución de las funciones se suman el valor de las 4 variables y si el valor es de 0, aprueba que el pipeline se ejecutó correctamente.

Control de calidad en los datos con la integridad en la base de datos relacional (por ejemplo, clave única, tipo de datos, etc.)

Para este punto se definieron 3 variables de calidad: calidad\_1, calidad\_2 y calidad\_3, cada una permite acceder a una función que cumplen diferentes tareas.

La primera validamos que no haya registros duplicados en las tablas transformadas por su primary key, en la segunda validamos conteo de datos de las dos fuentes y definimos criterios para dar aprobación y la tercera, comparamos el tipo de datos del dataframe crudo con el dataframe transformado y definimos reglas para que valide este item.

Si cumplen los criterios definidos en cada una de estas funciones, se retornará un 0, si no, retornara un 1, al final se realiza la suma de los 3 items y si el resultado es 0, lo definimos como aprobado.

Pruebas de unidad para los “Script” para asegurar que están haciendo lo correcto.

Para esta validación se define una función, donde escogemos dos datos particulares de la tabla cruda de la columna name y la columna price, estos datos son procesados y se genera la salida esperada en el tabla ya procesada, por ejemplo, el valor obtenido de la columna name de la tabla cruda, le quitamos todos los caracteres especiales y la transformamos en minúsculas, este dato obtenido sirve de filtro en la columna name de la tabla que paso por el pipeline, si obtenemos una coincidencia o más, es un ok, esto lo hicimos para la tabla listings\_edit, para la tabla calendar, obtenemos un valor especifico de la columna Price, este valor le quitamos su carácter especial y se vuelve a formato entero, este valor obtenido sirve de filtro en la tabla calendar que nos entrego el pipeline, y si hay una o mas coincidencias retornamos un ok. Se pueden hacer mas pruebas de unicidad como estas para el script pero por cuestiones de tiempo se crearon estas dos.

Comprobaciones de fuente/conteo para asegurar la integridad de los datos.

Este paso fue cubierto en las funciones de calidad de datos en el conteo de registros.

**EN EL REPOSITORIO ESTAN LOS SCRIPTS DE TODO ESTO.**

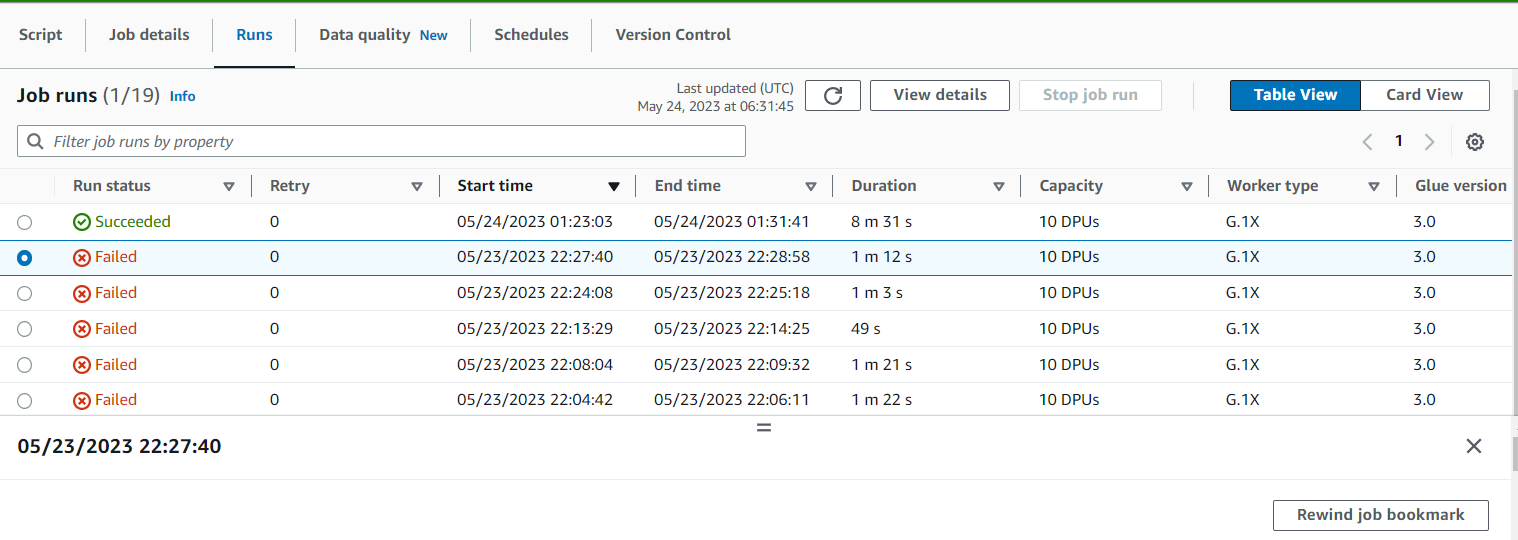
Finalizado todos estos controles y validaciones, si la base de datos procesada los cumple todos, podemos transferir nuestros datos a un entorno analítico como redshift.

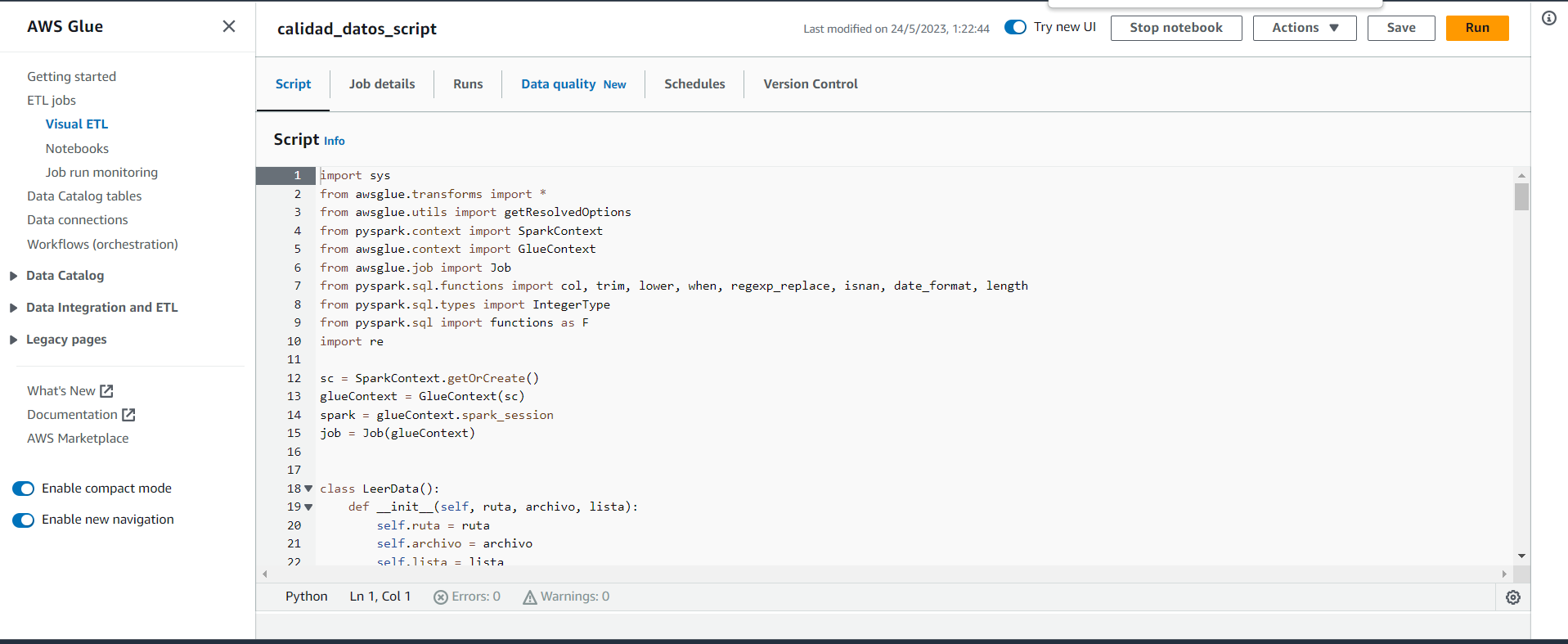
Zona analítica

Zona procesada

Zona cruda

A continuación, dejo una imagen de nuestro log de ejecución de controles extraído de cloudwatch y una imagen del script en glue



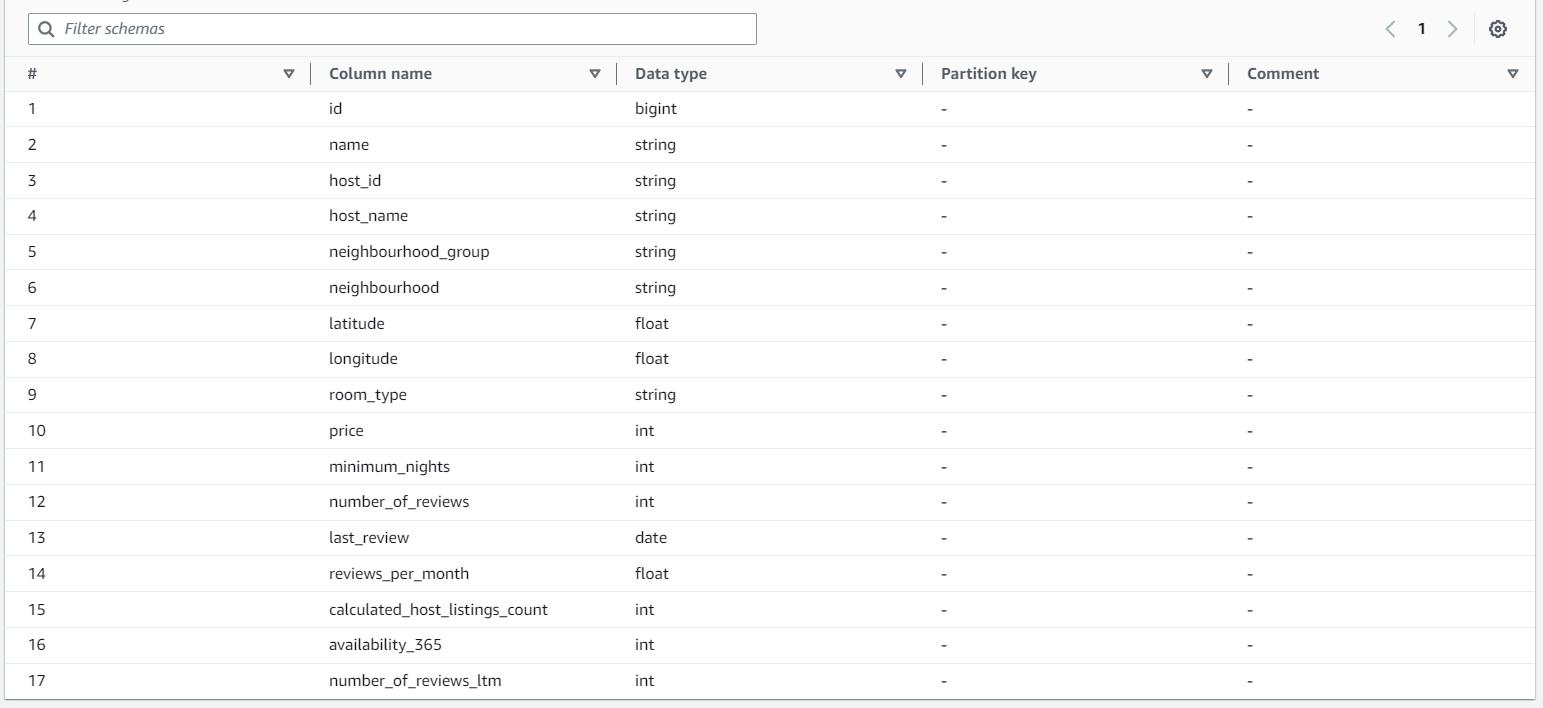


Los controles de salida se dejan en el repositorio

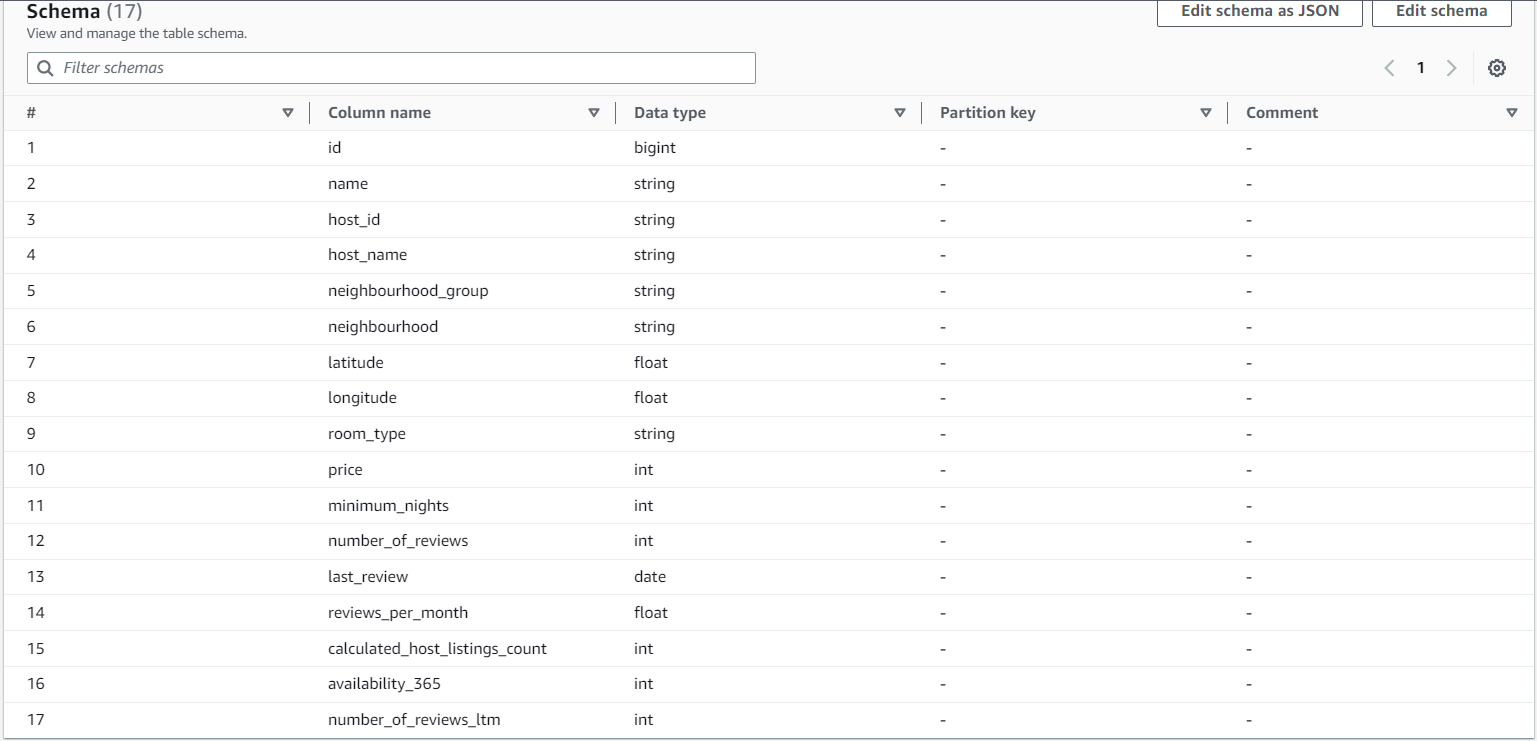
Incluir un diccionario de datos.

Para esto usamos un servicio de AWS glue (Data catalog tables) que nos entrega la metadata de nuestras tablas:

Listings\_edit\_trasnfo (procesada):



Listings\_edit (cruda):

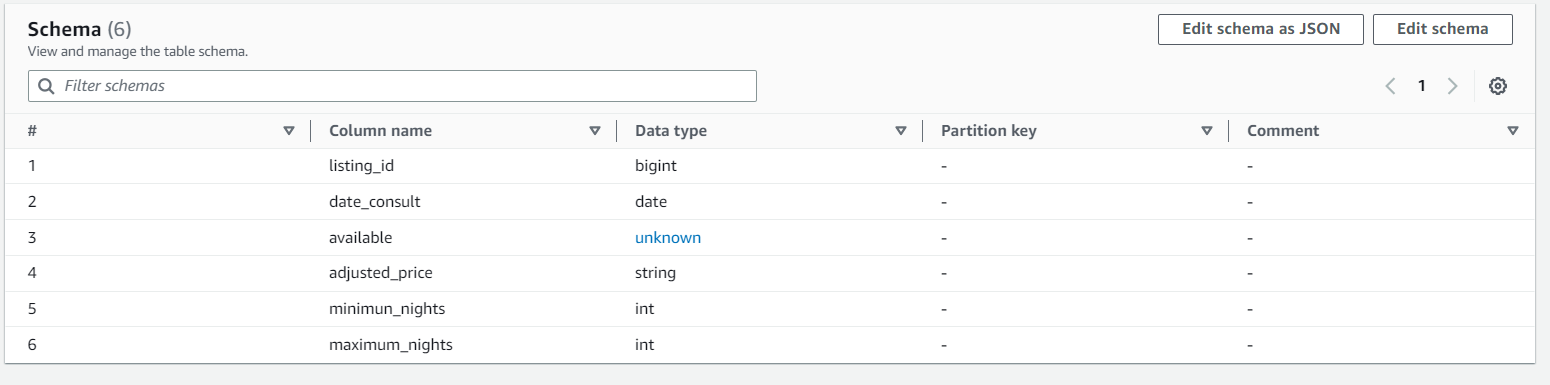


Calendar\_transfo (procesada):

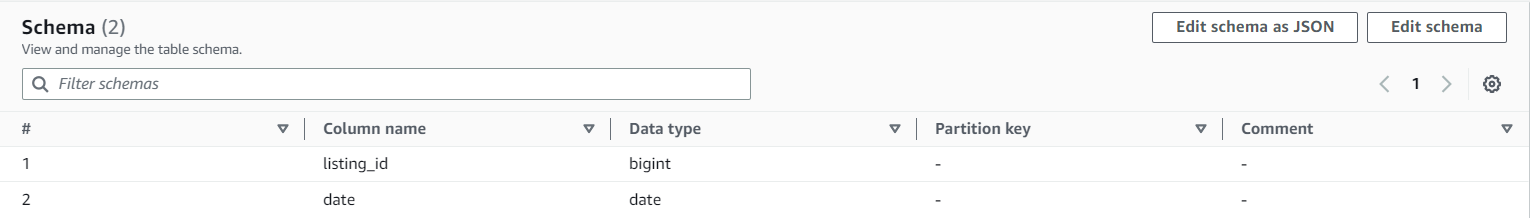
Tabla

Descripción generada automáticamente

Calendar (cruda):



Reviews (cruda):



Al momento de realizar exportaciones desde el script de glue, con ayuda de la librería hudi, se crean columnas extras que posteriormente se pueden eliminar.

Paso 5:

• ¿Cuál es el objetivo del proyecto?:

R/ Para mi el objetivo del proyecto es mostrar mis conocimientos en ingeniería de datos independientemente de la tecnología que se use, haciendo un proyecto desde cero empezando con conseguir la base de datos adecuada que en mi concepto es lo mas importante, hasta la entrega de una base lista para realizar analítica donde se puedan tomar decisiones en función de los datos.

• ¿Qué preguntas quieres hacer?

R/ ¿Cuales son las ventajas mas importantes de la tecnología en nube sobre las tecnologías on premise?

¿La tecnología on premise todavía es factible usarla como arquitectura principal?

¿Cuáles son las cualidades más importantes que debe tener un ingeniero de datos para trabajar en Nequi?

¿Al momento de hacer analítica es necesario tener los datos en streaming, o tenerlos en batch aporta lo mismo para el negocio de la banca?

• ¿Por qué eligió el modelo que eligió?

R/ Escogí el modelo dimensional estrella ya que es muy escalable, es fácil de interpretar, fácil de llevar a producción y genera mucho valor en los equipos de analitica de datos ya que es fácil de consumir.

• Incluya una descripción de cómo abordaría el problema de manera diferente en los siguientes escenarios:

o Si los datos se incrementaran en 100x.

o Si las tuberías se ejecutaran diariamente en una ventana de tiempo especifica.

o Si la base de datos necesitara ser accedido por más de 100 usuarios funcionales.

o Si se requiere hacer analítica en tiempo real, ¿cuales componentes cambiarias a su arquitectura propuesta?

R/ Gracias a la tecnología de la nube como AWS este incremento de datos no presentaría problema, ya que una de las características principales de esta tecnología es la escalabilidad, en caso de que se presente un incremento masivo de datos, lo ideal seria incrementar los recursos de los clusters de procesamiento de la arquitectura.

R/Si las tuberías se ejecutaran en una ventana de tiempo, lo que se puede hacer es vincular esta tuberia con un servicio de orquestación programable, que permita correr las tuberías en la ventana de tiempo que se parametrice, un servicio de estos puede ser AWS airflow.

R/Si la base necesita ser accedida masivamente, primero que todo se debe validar que las bases de datos sean lo suficiente optimas, usando técnicas como de particionamiento, o asignar roles específicos de consumo para las diferentes bases de datos, además de esto lo mas importantes es tener nuestras bases de datos en sistemas altamente escalables como aws redshift.

R/ La arquitectura actual no seria necesario cambiarla, pero si sería necesario agregar pasos previos para capturar los datos a medida que van llegando. Tuve la oportunidad de trabajar un tiempo en un proyecto donde la idea era construir un datalake en GCP con datos capturados en streaming, lo que se usó fue apache Kafka y Debezium como conector a las bases de datos productivas en MySql, si se quisiera implementar esto con servicios de nube, se podrían buscar servicios en AWS u otro Cloud que nos permitan replicar estas funcionalidades.