|  | Universidad Nacional Autónoma de México  Dirección General de Cómputo y de Tecnologías de Información y Comunicación |  |
| --- | --- | --- |

| Clave:  2022-12/6-1840 | Servicio Social:  Análisis de Datos Educativos |
| --- | --- |

| Supervisores:  Angélica Ma. Ramírez <  Miguel Zuñiga < |  |
| --- | --- |
|  | Alumno:  García García Juan David |

| Bitácora de Actividades  Fecha de inicio: 03/11/2022  Fecha de conclusión: Por definir |  |
| --- | --- |
|  | Proyecto:  Google Analytics V3  (Universal Analytics) |

El propósito de este documento es describir mis actividades a lo largo del proyecto, esto para documentar el análisis, investigación, implementación y toma de decisiones de los problemas presentados a lo largo de este proyecto.

La solución definitiva de los problemas se podrá encontrar en la documentación del proyecto, donde se abordará concisamente y servirá a modo de documentación oficial del proyecto.

**03 de noviembre de 2022:**

Fue el día de inicio de mi servicio social. Use el tiempo para la ambientación de mi equipo:

* Cambio de sistema operativo (Pop!\_OS a Windows 11)
* Ambientación de Windows 11 (Instalación de herramientas Open Source para trabajar con datos, por ejemplo, PostgreSQL, Python, Anaconda, Office 365 para estudiantes)
* Instalación de WSL2 con Ubuntu

Esto, tomando en cuenta sobre lo que se habló en la entrevista de introducción al servicio social, para optimizar el tiempo de acondicionamiento de mi equipo de cómputo.

**04 de noviembre de 2022:**

Se realizó una junta con los supervisores de mi servicio social dentro de las instalaciones de la DGTIC. En esta reunión, se me introdujo al primer proyecto en trabajar:

Familiarizarme con Google Analytics 3 (Universal Analytics) para descargar los datos de las distintas campañas de la DGTIC. Una vez hecho esto, revisar algunos temas con las direcciones IP para el buen uso de la herramienta y así evitar duplicados.

Una vez finalizada la junta, donde también se habló sobre cómo nos manejaremos, empecé a trabajar para resolver el primer problema.

En el tiempo restante, encontré algunas soluciones que podrían funcionar, así como las ventajas y desventajas de las mismas:

* Usar Google Looker Studio (antes Data Studio) para importar la información de los registros a un Dashboard y así poder descargar más de 5000 (cinco mil) registros.
  + Ventajas: Es gratuito y fácil de usar. Se puede replicar un solo Dashboard para todas las cuentas de Analytics. Se puede separar la información por mes o día con un botón de control.
  + Desventajas: Solo puedes descargar 50000 (cincuenta mil) registros por click. Existe un error (por revisar) donde el Customer\_ID no se registra en Data Studio, pero sí en Analytics.
* Usar una API de Python.
  + Ventajas: Teóricamente se podría descargar la información sin restricción alguna.
  + Desventajas: Las propuestas existentes dadas por la comunidad están desactualizadas (pues Google Analytics 3 está por ser obsoleto para dar paso a la versión 4), así como algunas librerías que usan. La propuesta de usar una API sería actualizar las existentes, o bien, crear una nueva desde cero. Crear una desde cero implicaría añadir un bot a los propietarios del proyecto dentro de Analytics.

**07 de noviembre de 2022:**

Se siguen explorando opciones para analizar cómo descargar más de 5000 registros por click.

* Usar Google Sheets mediante un Add-on.
  + Ventajas: Es gratuito y solo es necesario añadir los parámetros escritos para descargar la información.
  + Desventajas: Al igual que en Data Studio, existen columnas o dimensiones que se importan sin la información, pese a que en Analytics se muestra sin problema alguno.
* Usar Excel mediante un Add-on:
  + Es exactamente igual que en con Google Sheets, aún no se prueba esta opción porque el Add-on tiene un periodo de prueba muy corto.
* Usar Web Scrapping:
  + Es otra opción que aún no se ha intentando en implementar. Puede ser posible porque las tablas dentro de Analytics están construidas con HTML y no con una herramienta externa, “facilitando el trabajo”

Además de las opciones vistas, así como el tratar de solucionar los problemas con las soluciones pasadas, se crearon los documentos de bitácora y de documentación del proyecto. En caso de encontrar una solución definitiva, se dejará dentro de la documentación con una descripción detallada.

Parece que el id de los usuarios no puede ser exportado, ya sea por Add-on o por API por lo que esto restringe la información que puede ser recabada por estos medios.

<https://support.google.com/analytics/answer/6205850?hl=en&ref_topic=3123660#reporting&zippy=%2Cin-this-article>

Google te dejará descargarlos solo si eres propietario de Google Analytics 360, la cual es una versión de paga.

También se exploró el usar BigQuery para exportar el ID de los usuarios, pero esta opción también requiere de pagar (en este caso, servicio de Google Cloud)

Para finalizar el día de hoy, se está preparando una prueba para usar una API de Google mediante Python para ver si la información se carga correctamente. Entonces, se solicitará mañana el agregar el siguiente correo como lector:

[api-de-prueba@proyecto-prueba-367719.iam.gserviceaccount.com](mailto:api-de-prueba@proyecto-prueba-367719.iam.gserviceaccount.com)

También se volvió a usar el Add-on para Google Sheets y pese a que el ID de cliente no es importado, sí que se pueden importar el número de columnas y filas que el usuario guste, saltandonos el límite de Google Analytics así como el de Data Studio.

**08 de noviembre de 2022:**

El día de hoy se implementó un nuevo código escrito en Python. Se adaptó el código de muestra que se encuentra en la documentación de la API de Analytics; para esto se solicitó permiso de administración para añadir una nueva cuenta dentro de la página de Encuentro EDUCATIC, esto para experimentar con las pruebas.

El correo usado fue generado de forma automática dentro de un proyecto nuevo de Google Cloud, pues se necesitaba para la configuración de la API.

Las pruebas fueron un éxito parcial, pues si se puede descargar un reemplazo de el ID Usuario por uno de Cliente, este es generado de forma aleatoria pero conserva las características de llave primaria dentro de la tabla.

El problema surge con las propias limitaciones de la API, ya que no permite descargar más de 10 métricas por consulta, pero esto se puede solucionar automatizando las consultas y después juntar las tablas generadas mediante un JOIN.

Para el diseño de esta propuesta de programa, implementaré la automatización por mes (desde el primero registrado en Analytics hasta el último mes de ejecución del programa) para generar cada tabla de cada página de cada proyecto de Google Analytics.

Debido a que Pandas (la paquetería de manejo de datos que uso en Python) es lenta (en términos computacionales) para esta gran cantidad de información y operaciones entre tablas, propongo usar PySpark para aprovechar las bondades que Spark nos brinda al separar toda la información por nodos, lo que facilitará hacer el JOIN y otras operaciones entre relaciones (tablas).

Por último, dada a la característica del proyecto, así como las futuras propuestas que se puedan crear con la interpretación de esta información, creo óptimo dejar este programa dentro de un Jupyter Notebook.

El día de mañana empezaré a pedir el acceso con el correo de la API a las demás páginas de Google Analytics, esto en lo que empiezo a escribir el código de automatización de consultas así como la mejora del código original.

**09 de noviembre de 2022:**

En esta ocasión, empezaron las pruebas sobre importar información dentro de Google Analytics mediante la API conectada a Python. Lo primero que debemos destacar es que la API está limitada a 7 dimensiones y 10 métricas por consulta; el lado bueno es que las consultas son ilimitadas al igual que el número de filas que podemos extraer.

También, debido a la gran cantidad de información que se puede descargar mediante dichas consultas, se prefirió trabajar con PySpark definitivamente, por lo que el día de hoy se tomó parte del tiempo a instalar una versión de Spark estable dentro de mi equipo de trabajo. (Dicho tutorial de ambientación podrá encontrarse en la documentación del proyecto). Es importante destacar que, en caso de que se opte por usar herramientas basadas en la nube como Google Cloud (específicamente, Google Colab) el código también es productivo, evitando así configurar o ambientar cada equipo cada vez que el código se quiera correr.

Para finalizar las actividades del día de hoy, se empezaron a capturar algunos datos importantes que se encuentran en la página de desarrolladores de Google Analytics ( <https://ga-dev-tools.web.app/> ) donde están alojadas todas las métricas y dimensiones con las que trabaja Analytics. Para esta tarea, se seleccionaran las acordes para retraer la mayor cantidad de información no redundante, por lo que se proponen dos tablas:

* Audiencia: Servirá para almacenar toda la información por audiencia existente, se usará como llave primaria la ID de cliente (clientID). Dicha clave es distinta a la clave de cliente de Analytics (customerID) así como la de usuario (userID), pero sigue siendo única lo que servirá para identificar de cierta manera a cada usuario.
* Detalles por página: En este caso, se guardarán todos los detalles relacionados por cada página web que integre el sitio, no por usuario único. Por ello, la clave primaria será la misma página.

**10 de noviembre de 2022:**

El día de hoy se continuó con las pruebas para verificar el límite de la API. Después de realizar algunas pruebas de rendimiento, se reconsideró usar Pandas en vez de PySpark para la manipulación de los datos arrojados por la API, pues el tiempo de trabajo entre ambos métodos no es muy distinto. La desventaja es que parece que Pandas ocupa ligeramente más memoria RAM.

Esto todavía está en consideración, pues la especialidad de PySpark está en que los datos sean masivos, lo cual aún no está definido por completo. Se entiende que son diversas fuentes y proyectos de Google Analytics, pero aún no hay certeza si la cantidad de datos a descargar es grande (en términos de GB de información). Por ejemplo, al descargar la información de 300 días de un proyecto este llegó a pesar poco más de 27 MB.

También se catalogaron en formato CSV la información más importante que se tiene que rescatar, así como la información por proyecto de Analytics. Con esto en mente, podemos editar fácilmente en dichos archivos los parámetros necesarios a buscar para evitar la saturación del código con dichos datos, se pueden ver como una especie de archivo de configuración necesario para el programa.

Por último, el día finalizó con otras pruebas más sobre el uso de la API. Se encontró un bug dentro de la API (No del código hecho en Python) donde arroja un error dentro de la misma sobre la cantidad de dimensiones a añadir, dejando las teóricamente siete dimensiones a solo una. Otro problema que se encontró dentro de la misma es que se puede romper al ejecutar consultas con grandes días de separación. Esto se pudo observar al momento de solicitar toda la información recabada del sitio (desde su fecha de creación hasta la actualidad), al cabo de unos minutos la conexión se rompió, esto por parte de la API.

Ambos problemas se pueden solucionar automatizando las consultas de tal forma que se haga una consulta por cada dimensión con un periodo de tiempo no mayor a un año. Esto se podría hacer fácilmente usando algunas funciones y algunos ciclos, por lo que se está trabajando actualmente en ello.

**11 de noviembre de 2022:**

El día empezó con una junta, la junta habitual de los viernes. Esta vez, por algunos problemas de tiempo, se llevó a cabo a las 14:00 horas y se propuso trabajar la siguiente junta a las 11:00. Se mostraron algunos avances de lo trabajado esa semana, así como la propuesta y el giro que iba a tomar el proyecto, por lo que se decidió dejar la API mediante Python ya que se encuentra en una fase algo, relativamente, avanzada.

Una vez que la junta se terminó, se siguió trabajando con la API. Se tomaron en cuenta algunas consideraciones debido a los recursos disponibles dentro de la oficina, por lo que se deberán de hacer diversos ajustes, ya que el programa actual puede llegar hasta los 20 GB de memoria RAM en momentos críticos del mismo, debido a la alta cantidad de información manejada.

**14 de noviembre de 2022:**

El día de hoy se trabajó con la parte de optimización del programa. En las pruebas, como se mencionó anteriormente, Pandas puede llegar a ocupar hasta 32GB de memoria RAM en momentos puntuales, por lo que es inviable seguir con el programa actual, ya que al alcanzar el límite (en mi caso, 24GB) el programa se detiene con un error y la información no puede tratarse.

En lo que se piensa una forma dinámica para resolver el problema, se decidió empezar con otra tarea más sencilla, por lo que se empezaron a elegir las métricas y dimensiones necesarias para generar dos fuentes de datos: usuario y web. Se empezará con usuario, por lo que se probaron diversas métricas y dimensiones. Para mencionar algunas cosas curiosas, pedir métricas como “Dispositivo Móvil” rompe con la explotación de información debido a que usuarios no móviles se ven discriminados y eliminados del informe. Lo mismo pasa con características como Edad o Género, debido a que esta información no se encuentra disponible o configurada dentro del proyecto de Analytics.

El día finalizó con las pruebas de estas métricas, dejando para mañana el diseño de requerimientos para la explotación de la información.

**15 de noviembre de 2022:**

El día de hoy se empezó el diseño sobre el mismo código para pedir la información en el menor número de consultas pero sin perder información. Se encontró que Google Analytics permite descargar el máximo de información si se le piden dos dimensiones: sesiones y ID de cliente, por lo que ambas estarán repetidas en cada consulta.

Una vez que se experimentó con eso, tocaba separar las dimensiones en paquetes de información para evitar la eliminación de columnas, quedando como 6 consultas por fecha por proyecto por vista de Analytics. Ahora, al tener 6 reportes, solo es necesario leerlos como DataFrame con Pandas y hacer un Join por cada archivo usando el clienteID como llave primaria. Las columnas repetidas solo se eliminan antes de hacer el Join y tenemos como resultado un solo reporte por fecha.

El día de hoy se cerrará con la optimización del código, pues en periodos de 150 días el programa tiende a consumir las enormes cantidades de RAM mencionadas en días anteriores. Con esto en mente, el día termina con la optimización del mismo código, eliminando variables y haciendo pruebas para el número de días, ya que si se reduce demasiado, el programa se rompe debido a un error propio de la API y no del programa, ya que se “rompe” con enormes cantidades de consulta por un periodo de tiempo determinado.

**16 de noviembre de 2022:**

El día de hoy fue algo complejo, pues una vez que se decidió sobre qué métricas y dimensiones son útiles para descargar los datos de usuario, ahora se tiene que lidiar con la optimización del programa.

El problema surge con el manejo de recursos dentro de Pandas y es que, pese a no ser una herramienta que demande recursos del procesador, si es una que use la memoria RAM de forma constante y demandante. Se hicieron diversas pruebas dentro de Pandas, descargando toda la información de los usuarios y haciendo las diversas operaciones entre DataFrames requeridas y en ciertos puntos sobrepasó los 30 GB de RAM de forma teórica (esto según el mensaje de error, que decía que no podía almacenar toda esa información en 24GB que tengo actualmente).

Se decidió optimizar el código eliminando variables, guardando valores dentro del código y no de archivos externos para evitar gastar RAM, también se probó eliminando un poco de versatilidad dentro del código y quitando variables dinámicas, pero no hubo mejoría. En el mejor de los casos, el programa se rompía al hacer el Join de toda la información.

Para finalizar el día, se propuso reescribir el código y transformar el proceso de explotación de información: en vez de requerir la información por secciones pero desde el día cero hasta el presente, ahora se requiere la información completa pero seccionado por fechas.

El día de mañana se realizarán pruebas para ver si este método es mejor que el anterior, o bien, si se tendrán que usar herramientas externas a Python (como PostgreSQL, que hacer operaciones tipo Join entre tablas está mucho mejor optimizado que en Pandas) o incluso si se retoma el uso de PySpark, que ahorraría recursos pero aumentaría casi al doble el tiempo de operaciones.

**17 de noviembre de 2022:**

Después de realizar algunas pruebas, el método de reducir el intervalo de fechas funciona, pese a que el tiempo de procesamiento aumentó considerablemente (solo el tiempo, el uso siempre se ha mantenido en menos del 10%, esto para un procesador Ryzen 5 de 6 núcleos y 12 hilos).

El problema es que siguen existiendo segmentos puntuales donde la memoria RAM reporta hasta picos momentáneos de 13 GB, esto cuando el uso del sitio en determinado tiempo es grande. Esto podría reducirse al acortar aún más el tiempo de consulta por periodos de fechas, pero a mi consideración me parecería excesivo, pues el periodo con el que actualmente se trabaja es más que suficiente (30 días por cada consulta). Tomando en cuenta que existen sitios con bastantes años, el reducir las fechas impactaría negativamente en la velocidad del programa (pues por cada falla de la API se debe de esperar 30 segundos para volver a generar consultas, muchas consultas implican un número más grande de fallas de la API y con ello, más tiempo).

También, otra optimización añadida fue guardar en memoria “caché” dentro del disco duro cada DataFrame en formato csv, para después borrarlo dentro de la ram y una vez que todas las consultas se hiciesen, leerlos y hacer operaciones tipo Join y Union All entre ellos, para guardar en csv el documento final.

También, existen sitios que no contienen vista alguna (por ejemplo, becarios con id 60837979) por lo que se implementó un manejo de error donde se genera un archivo txt con el título del sitio diciendo que no existe información alguna.

Para finalizar el día, se dejarán descargando todos los conjuntos de datos con estas implementaciones, esperando que no ocurra algún error.

Los archivos relacionados con los usuarios quedarán guardados en la carpeta de “produccion/usuario” bajo la siguiente notación:

*Cuenta\_de\_Analytics-Propiedades\_y\_aplicaciones-Vista-ID.csv*

Después de dos horas de que el programa corriese activamente, se descubrieron algunos detalles interesantes:

* Debido a que existen periodos de actividad muy altos, algunos días se separarán en periodos de tiempo más cortos, para así recabar toda la información posible, pues la API no envía los elementos de los últimos 30 días.
* Por la razón anterior, se encontraron puntos donde se alcanzaron los 20 GB de ram de uso.

También, cabe destacar que el tiempo es un enemigo más pues esta sesión de poco más de dos horas, solo se pudieron hacer 8 páginas de Analytics, donde las dos más pesadas tienen archivos de más de 3.5 GB de almacenamiento, superando así las 10 millones de filas.

Tal vez el problema del almacenamiento no sea tan grave, pero se propone cambiar de 30 días a 15 días, por lo que cada consulta recuperará información de un periodo de 15 días.

**18 de noviembre de 2022:**

En esta ocasión, el día empezó con la habitual junta de los viernes. Se informó sobre los avances relacionados a la semana pasada y a todo le fue dado un visto bueno, también se pensó que, a futuro, una de las ideas para almacenar la información para su posterior análisis y limpieza, sería utilizar una base de datos como PostgreSQL para facilitar el trabajo.

El resto del día se dedicó a dos cosas, encontrar un error que provocaba pérdida de información así como el ver tutoriales de instalación de PostgreSQL en Windows, esto para estar preparados en caso de tener el visto bueno con los siguientes pasos.

**22 de noviembre de 2022:**

Este día se enfocó en optimizar el código ya existente, esto porque el viernes pasado ví que tenía algunos problemas además de la pérdida de información. Lo primero que se arregló fue el Join, pues había aparente pérdida de información (la más notable, las marcas de teléfonos o sistema operativo).

Después de diversas pruebas, se optó por dejar un outer join, por lo que no habría pérdida de información pese a que esto permitiría que las columnas se duplicarán, por lo que se empezó a trabajar en otra parte del código para eliminar este nuevo problema.

Al final, el día cerró con las pruebas del nuevo código, dejando descargando las consultas durante un tiempo para ver si realmente se llegaron a las optimizaciones deseadas en el producto final.

**23 de noviembre de 2022:**

Después de revisar algunos resultados finales, se compararon con los archivos csv anteriores y se encontraron que los nuevos no tenían la cantidad de información repetida como sí lo era anteriormente, también, esto provocó que el peso de los archivos se redujese considerablemente, pasando de los más pesados de 3.5G GB a solo 2.6 GB.

También, se empezaron las pruebas con un nuevo error que Analytics presenta después de largos tiempos de consulta. Una vez que se investigó el origen de este error en la documentación, se concluyó que no se puede hacer mucho para detenerlo porque surge directamente dentro de la API.

Por decir un ejemplo, los errores sobre la conexión inestable, el resultado de consultas nulas o la interrupción de comunicaciones con la API tienen solución dentro del código pues solo bastaría generar una consulta diferente o reintentarla, pero este error no puede tratarse de la forma anterior porque Python no puede interpretar el error y pese a los múltiples intentos de excepciones implementados, siempre se encuentra una nueva forma de saltarlos y romper el programa.

Después de intentar otras cosas más, se concluyó con que no es manejable, por lo que se implementó una especie de “reinicio”, donde se le pedirá al usuario el ID de la página en curso, corriendo nuevamente el programa.

**24 de noviembre de 2022:**

Una vez que se estableció que el código generado por usuario es productivo, al menos en mayor parte, se empezó a buscar las métricas y dimensiones necesarias para generar el segundo reporte: el reporte por página web.

Este reporte servirá a modo de descripción básica del rendimiento de las páginas que conforman el sitio web, después de hacer diversas pruebas, se exportaron las métricas útiles para este caso.

De igual forma, el día se cerró con el añadido de nuevas métricas para los reportes de usuario, por lo que los reportes generados hasta este momento quedarán obsoletos, lo cual no es un problema grave dado que la nueva información es bastante útil, como lo es la página por la cual se entró o el número de páginas por las que se navegaron.

**25 de noviembre de 2022:**

Este viernes no hubo junta, pues la semana tuvo una duración más corta de lo normal y debido a que existían juntas en los horarios propuestos, por lo que se moverán al lunes de la siguiente semana.

El día de hoy se implementó el código de los reportes por página web al código original, para aprovechar la automatización implementada por usuario. Esto se logró adaptando y modificando el código, por lo que ahora ya se pueden general consultas de ambos con la casi nula participación del programador.

Una desventaja de añadir nueva información es que ahora los archivos, que ya estaban optimizados, empeoraron de estado, pues ahora los nuevos campos importan información como la URL del sitio, aumentando bastante el tamaño de los mismos, teniendo cosas como que un archivo de 1 GB pasara a uno de 10 GB. El día de la junta se mencionará este hecho para ver cómo proceder o si es necesaria esa información.

Por último, se corrigieron los errores que la acción anterior generó, pues el código para los reportes del usuario se rompió en algunas partes. Después de repararlo, se iniciaron las pruebas finales y todo va bien, tomando en cuenta que es normal que los reportes de usuario se tardan y que los reportes generados por página requerían más métricas de las tomadas en cuenta, por lo que ya podría quedarse productivo. Por último, si todo va bien, se tiene planeado empezar a escribir la documentación la siguiente semana.

**28 de noviembre de 2022:**

El día de hoy tuvimos una junta, para presentar los resultados de lo trabajado durante la semana pasada. Se planificaron los siguientes pasos a seguir, por lo que se confirmó que ahora tocará mover los archivos generados a una base de datos (en este caso, PostgreSQL).

Ahora, debido al cambio del nuevo objetivo, se empezó a modificar el programa original, esto para ahorrar tiempo de procesamiento y recursos ya que, dentro de Postgre, se llevará a cabo la normalización de la información así como el cruce de información por tablas.

El día concluyó con las pruebas de la nueva versión del código, donde ahora se pedirá la información en distintos documentos csv por categoría y por fecha.

**29 de noviembre de 2022:**

Una vez que se comprobó que la información consultada es correctamente entregada, ahora toca solucionar el nuevo problema generado por la enorme cantidad de CSV generados. Para esto, se propone usar algunas herramientas ETL para la concatenación de documentos.

Para lograr este objetivo, empezamos con la normalización de los lugares donde se almacena la información, creando dos carpetas por reporte: usuario y página.

Por el lado de los reportes por página, el programa crea sin problemas los reportes enteros, es decir, juntando la información por fechas y cruzando las tablas usando como ID la página del sitio.

Para los reportes de usuario, se ubicó que el problema de uso excesivo de memoria RAM era dado por el cruce de las tablas usando como llave primaria el ID de Cliente, dicho ID generado de forma aleatoria y única por Google Analytics. También, otro problema causado por la forma de la construcción del código y por la API de Analytics, el tiempo de consulta era alto debido a que dicha API tiene un límite, por lo que la suma de estos factores hacen lento el proceso.

Por ello, se programó una nueva parte donde ahora se elimina el cruce de las tablas, ahorrando el consumo de memoria RAM y manteniendo el uso (contando el sistema operativo y dos ventanas de Chrome) a 5.5 GB de RAM. Y con la automatización generada por el concatenar documentos se pasó de tener por cada página un documento CSV a tener 7, pasando de archivos que pesan 10 GB a archivos que, sumando, pesan menos de 1.5 GB.

**30 de noviembre de 2022:**

El día de hoy se empezó la ejecución del nuevo programa (versión 1.5), generando los siete archivos csv por usuario y uno por página web. Pese a que no requiere mucho esfuerzo, se tienen que supervisar el proceso debido a los errores que pueden ocurrir durante el proceso (errores heredados por la API y no por el programa, ya que estos se encuentran debidamente manejados con excepciones).

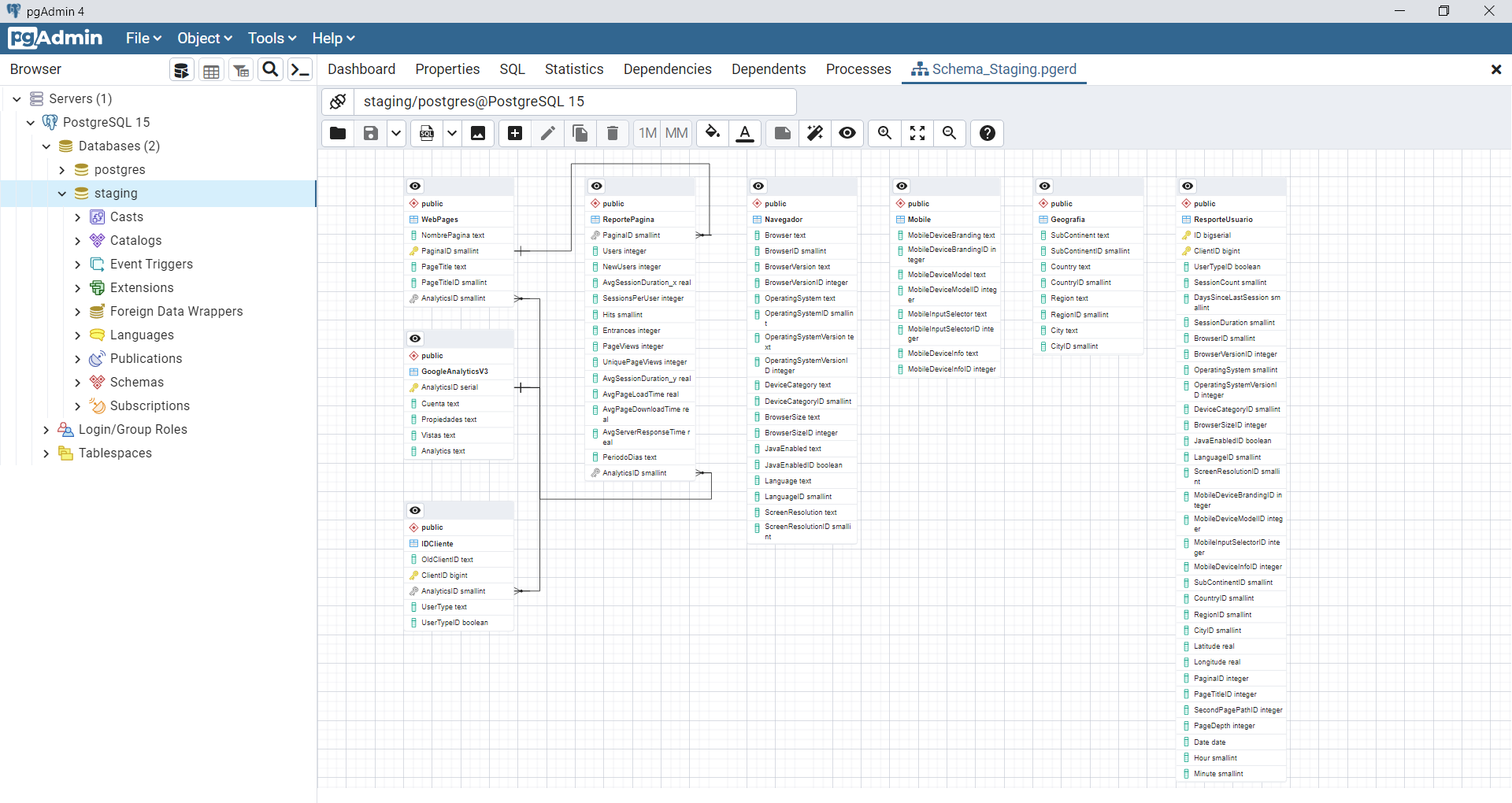
El programa resultó con un problema durante todo el tiempo de ejecución, en la página número 28 de 49, debido a que la cantidad de información consultada es tan grande que la API “moría” en el proceso, generando el error *timeout*, que en resumidas cuentas informa que la API sobrepasó el tiempo de espera predefinido por Google.

El día terminó con la generación de 33 registros, por lo que se continuará con el proceso al día siguiente.

**1 de diciembre de 2022:**

Una vez que se descargaron los demás registros, se empezó con la ambientación del equipo para la implementación de la nueva base de datos. Se empezó instalando PostgreSQL en el equipo, usando las preferencias predeterminadas, por lo que se realizó la instalación habitual.

Una vez instalado, se empezó a crear el esquema de la base de datos, utilizando la herramienta *ERD* de *pgAdmin*, esto para tener el entorno de desarrollo en un solo lugar. También se creó la base de datos que servirá para almacenar las nuevas relaciones. El diagrama quedó de la siguiente manera:



**2 de diciembre de 2022:**

Como el programa, en principio, ya no recibiría modificaciones, se empezó a escribir la documentación del mismo. Dicha documentación será un documento que contendrá la información del programa escrito en Python y probablemente también tenga la documentación de la base de datos SQL. (Esto último, estaría por definirse en la siguiente junta).

También, se aprovechó el día para subir la información consultada y capturada de la API a una carpeta de Drive, en dicha carpeta se encontrará la información capturada con la versión 1.4 (obsoleta) así mismo como la de la 1.5 (la actual).

**5 de diciembre de 2022:**

El día empezó con la junta habitual de los lunes, mostrando los avances de la semana pasada. También se trataron temas relacionados con el futuro del proyecto, donde se confirmó que la documentación estará unificada en un solo documento, así como que el esquema presentado se usará para el diseño de la base de datos (incluido el unir todos los informes de Analytics por reporte en una sola tabla para cada uno).

Debido a esto, se presentó una nueva modificación del programa que conecta con la API (llegando así a la versión 1.6) ya que se encontraron algunos patrones que indican pérdida de información con respecto a los reportes generados por página. Se empezó a modificar el código de la función de reporte por página, teniendo como objetivo dos archivos por cada vista de cada página del Analytics.

**6 de diciembre de 2022:**

El día de hoy se siguió con la implementación del código, una vez que se encuentra operativo se empezaron las descargas de los reportes, teniendo un aumento considerable de líneas generadas por consulta, así como el peso de los mismos archivos.

El día finalizó con el diseño de un nuevo programa escrito en Python que ayude a imprimir los valores únicos de las columnas que tienen datos repetidos, por ejemplo, el navegador o el sistema operativo.

**7 de diciembre de 2022:**

Empecé a escribir el código para obtener estos valores únicos, pensando en simplificar los procesos lo máximo posible, por lo que evité el usar funciones para hacerlo todo “más directo”, por lo que se optó por automatizar todo con ciclos while para cada objeto dentro del CSV llamado ID.

El día terminó con una propuesta de código (no la versión final) ya que el proceso de filtrar los datos es tardado y en algunos casos puede consumir más memoria RAM de la deseada.

**8 de diciembre de 2022:**

El día de hoy se terminaron de hacer los ajustes correspondientes al programa anteriormente mencionado, esto logrando el objetivo de optimizar tiempos y garantizar que los valores otorgados como resultado sean verdaderamente únicos.

También, se empezó a escribir el script de SQL tomando en cuenta la estructura propuesta en días anteriores. El día finaliza con el inicio de este script, donde se tomará en cuenta los archivos en formato csv con los valores únicos para la creación de las tablas auxiliares que evitarán la recurrencia de valores por columnas.

**9 de diciembre de 2022:**

El día inicia con la escritura del script de SQL empezado el día de ayer, también, se está empezando la escritura y continuación de la documentación.

El día finaliza con la propuesta de una nueva esquematización de las tablas auxiliares, con el propósito de clarificar la indexación de valores únicos de una mejor manera, para evitar posibles ambigüedades.

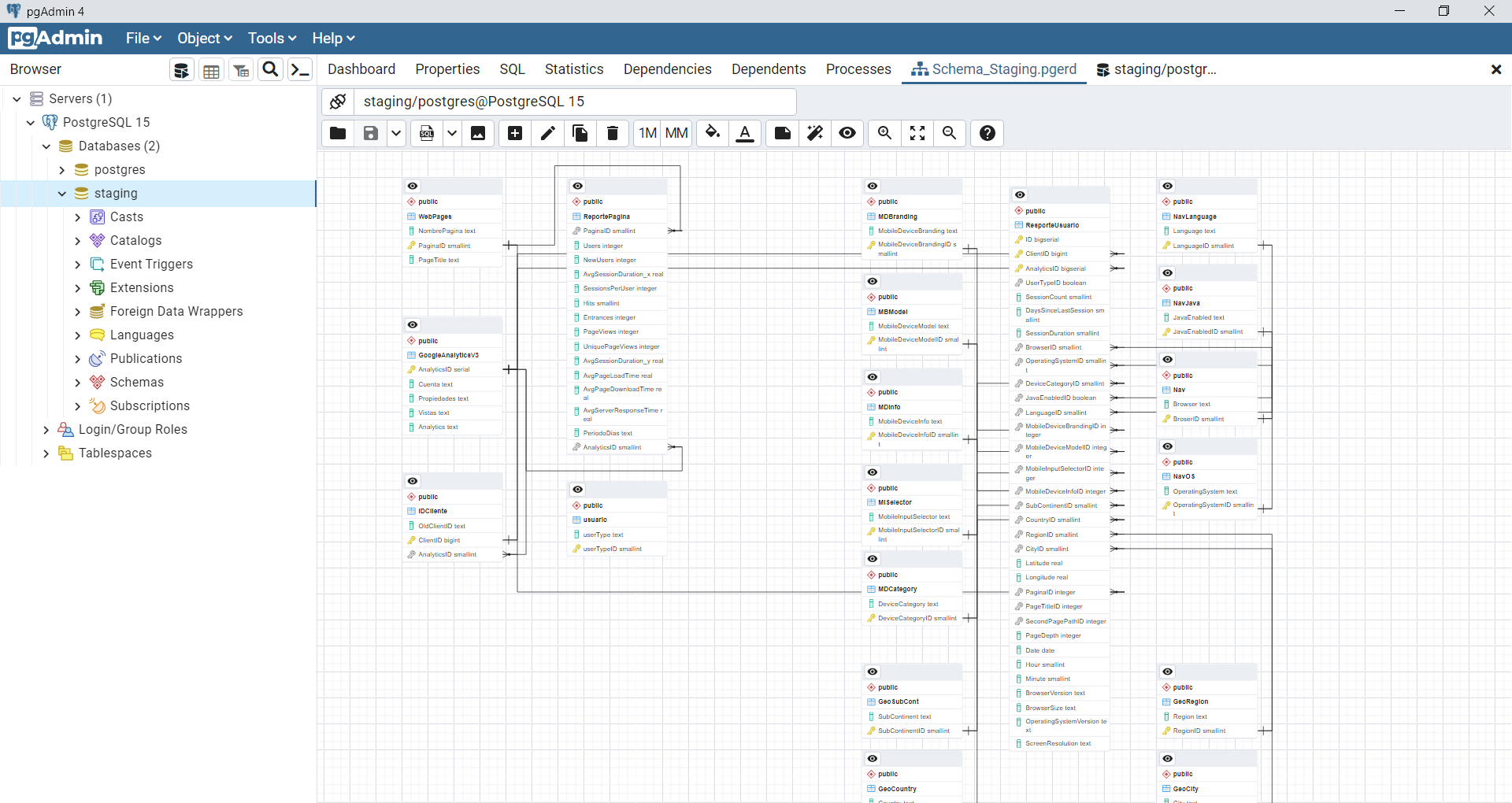
**5 de enero de 2023:**

Se empieza la construcción de la esquematización de la nueva propuesta. En este caso, se usarán más tablas para evitar ambigüedades o pérdida de información, ya que algunas operaciones dentro de PostgreSQL entre tablas con la arquitectura anterior causaban la importación de datos erróneos.

El día se termina con el inicio de la propuesta de esta nueva arquitectura dentro de un archivo ERD dentro de PostgreSQL.

**6 de enero de 2023:**

Este día se dedicó por completo a la finalización del esquema ERD dentro de PostgreSQL, solucionando los problemas causados por la primera versión, así como la implementación de la nueva arquitectura que consiste en la creación de una tabla por cada característica que los reportes de usuario o página puedan tener, quedando así:



Esta nueva implementación provocará que el archivo script SQL en el que se trabajó en los últimos días hábiles de diciembre quede obsoleto y por tanto, no cumpla con las nuevas necesidades de este esquema. Por lo tanto, se planea crear uno nuevo, aprovechando las bondades que esta arquitectura trae consigo.

**9 de enero de 2022:**

Se tiene planeado trabajar durante toda esta semana en el script SQL, que tendrá como propósito la ingesta de información de los archivos CSV generados en Python dentro de una base de datos relacional con la ayuda de PostgreSQL, esto implica que el script se encargará de hacer las operaciones de limpieza y extracción de datos de forma automática.

Por lo que, el día de hoy empezó con la lectura de parte de la documentación de PostgreSQL, así como la búsqueda de información en línea, pues se encontró que existen algunas formas de abordar el problema. En particular, se buscó sobre cómo ejecutar texto plano en forma de sentencias SQL así como la declaración de variables locales dentro del script, también se buscó cómo darle permisos de lectura de PostgreSQL para evitar errores.

**10 de enero de 2022:**

Una vez que se experimentó con la implementación de la información obtenida el día de ayer, se empezó a escribir el script. Se tomó la decisión de escribir todo lo necesario en un solo documento, en vez de escribir una serie de documentos más pequeños, ya que se planea agregar la descripción del comportamiento del script dentro de la documentación, aunado a que se planea escribirlo de forma universal, es decir, con el propósito de que pueda ejecutarse en cualquier equipo con la misma versión de PostgreSQL o superior, los archivos CSV y un cambio ligero en el script, que consiste en añadir la ruta de ubicación de dichos documentos.

El día finalizó con la escritura de la creación de las nuevas tablas a usar, así como la propuesta de implementación de la ingesta de los archivos CSV al tener luz verde con las pruebas realizadas.

**11 de enero de 2022:**

Este día se enfocó en la implementación de la información guardada en los archivos CSV de valores únicos a sus respectivas tablas. Para ello, se escribieron todas las variables necesarias para realizar todo el proceso que se tiene en mente dentro del script, para luego pasar a la ingesta de información y limpieza para el almacenamiento de estos valores únicos.

El día terminó con la corrección del código escrito previamente, esto para hacer más eficiente la indexación de los valores únicos, ya que en el código escrito recientemente se insertaban los valores ID directamente del CSV empezando por cero dentro de un formato INTEGER, pero ahora se cambiaron a formato SERIAL y se generan automáticamente, permitiendo así que PostgreSQL genere una indexación mucho más eficiente y por tanto, se disminuyó el tiempo de ejecución en entorno de prueba en 10 segundos.

**12 de enero de 2023:**

Para optimizar el script desde el principio, se optó por usar funciones que se encargaran de crear y de borrar tablas temporales, en las cuales se estarán haciendo las operaciones correspondientes para dejar los datos productivos e indexados según la arquitectura propuesta.

No se llegó a mucho, pues después de algunas horas de investigación, por alguna razón la creación de las funciones falla al ejecutar el script (pero no lo hacen si se crean en uno aparte), lo cual trae un dilema pues si se requieren usar se debería de crear un archivo aparte, lo cual va en contra de lo propuesto, pero si no se hacen, se tendría que agregar el código para crear y eliminar dichas tablas, que en principio no afectaría el rendimiento del mismo, pero que sí generaría código redundante, faltando así a las buenas técnicas de programación.

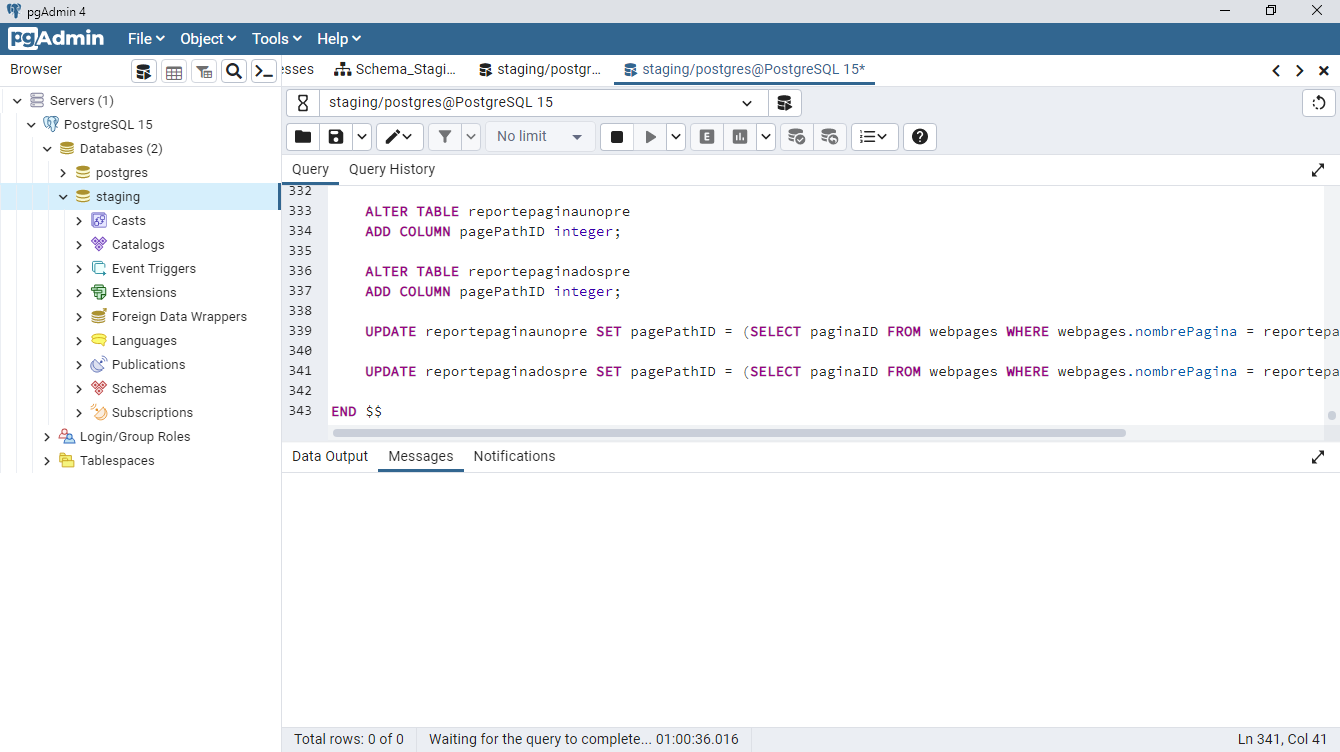
**13 de enero de 2023:**

Después de investigar sobre el tema de las funciones, se optó dejar el código redundante de creación y eliminación de tablas temporales, esto por facilidad de ejecución en otros equipos, ya que no afecta el rendimiento y hace que sea mucho más fácil ejecutar el script a que si este depende de otros.

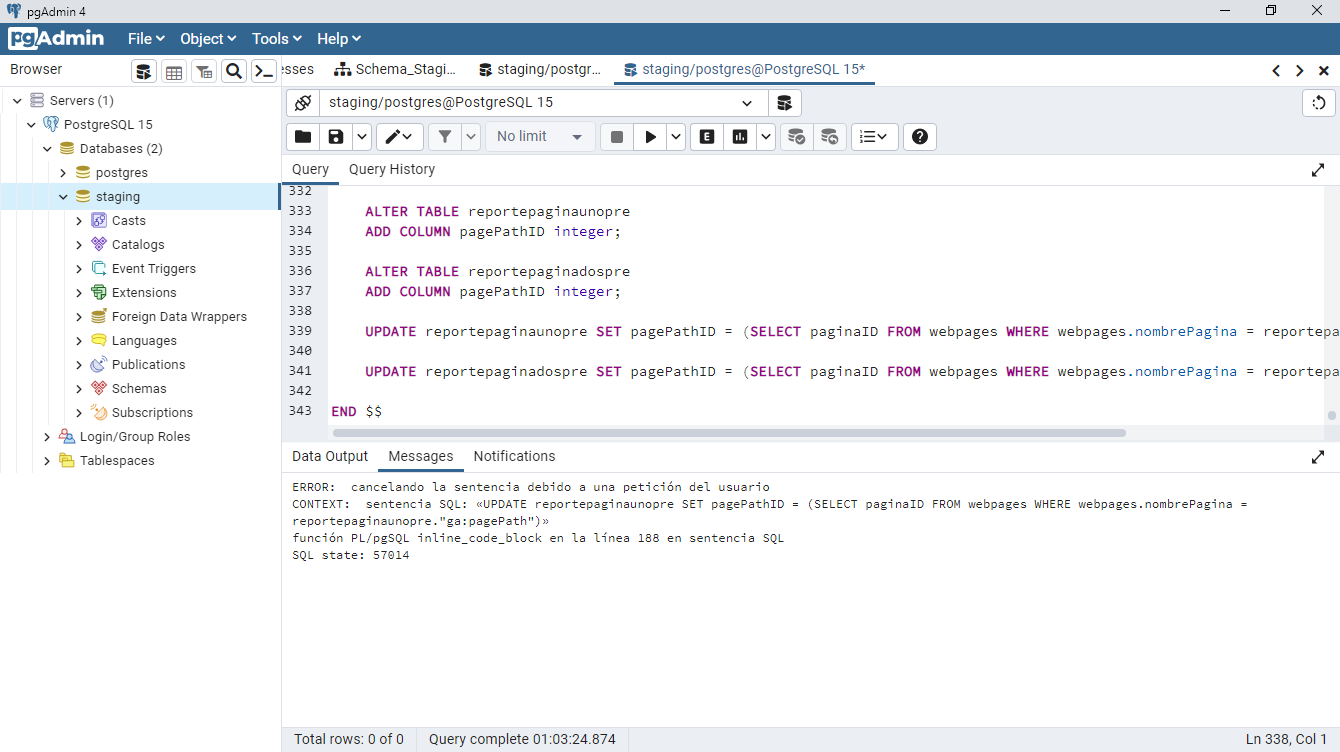
También, se iniciaron las pruebas de limpieza en reportes de página. En este caso en particular, se empezó por la asignación de un ID único para cada valor del reporte de páginas, esto para hacer mucho menos pesado el tamaño de las bases de datos y de paso optimizar las operaciones entre tablas, ya que es mucho más fácil hacerlas mediante un número entero que mediante una dirección web completa.

El problema que no se ha podido solucionar es que la operación de filtrado del primer reporte ha tardado más de una hora (en teoría, se interrumpió después de una hora porque el proceso no se había terminado), lo que es inviable tomando en cuenta que es la primera limpieza de un reporte de más de veinte, solo de las páginas web (pues los de usuario presentan mayor complejidad).

Se anexar evidencia del caso:



Una vez que se interrumpe, nos arroja el error de que nosotros cancelamos la consulta, así como en la línea que se quedó trabajando:



Por lo que se trabajará en una nueva forma para asignar los ID correspondientes para la introducción de dichos datos. Se tratará de llevar a cabo mediante la programación del script tipo PL/PgSQL, pero si no se llega a un avance óptimo, se optará por retomar Python como un posible candidato para la limpieza, dejando a PostgreSQL como el almacenamiento de los datos procesados.