Eugenio Ribón, Jorge Kindelán

INFORME PROYECTO

17 de abril del 2024

VISIÓN GENERAL

Como se ha descrito en el póster del proyecto, el proyecto trata de manera transversal distintas bases de datos, tanto relacionales como no relacionales. Dividido en distintos módulos que van desde el preprocesamiento y carga de los mismos en las distintas bases, la obtención de información de estas mediante “queries”, hasta su visualización en una interfaz gráfica interactiva.

OBJETIVOS

1. Cargar de manera eficaz los datos de reviews de amazon, dividiéndolos entre: sql (base de datos relacional) y mongodb (base de datos no relacional).
2. Obtener información de las bases de datos de manera interactiva para el usuario y mostrarlas en una interfaz gráfica
3. Almacenar ciertos datos en neo4j, siguiendo las consultas elegidas por el usuario, mediante un menú sencillo.
4. Mostrar las visualizaciones en otra plataforma como PowerBI o Tableau y pensar un pequeño modelo de machine learning para recomendaciones.

ESPECIFICACIONES

Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto.

RETOS DURANTE EL PROYECTO

PRIMERA PARTE: DISEÑO Y CARGA DE DATOS

Este proyecto comenzó con el reto de plantear el diseño de la base de datos, siendo necesario combinar sql con mongodb, tras plantear varias soluciones a este problema llegamos a la conclusión de que almacenaríamos toda la información que consideramos más “estructurada” en

sql, incluyendo aquí el reviewerID, asin, overall, unixReviewTime, reviewTime, category, reviewerName. Esta información la almacenamos mediante una estructura de tres tablas: Una para los reviewers donde almacenamos el reviewerID (Primary key y Foreign key en la tabla de reviews) y reviewerName, una tabla para los productos donde almacenamos el asin (Primary key y Foreign key en la tabla de reviews) y la categoría y por último, una tabla de reviews, en la que generamos su propio reviewID (Primary key y relación directa con mongodb), reviewerID (FK de reviewers), asin (FK de products), overall, unixReviewTime y reviewTime. Por otro lado en mongo almacenamos el reviewID (Primary key que se relaciona con la de sql), helpful, reviewText, summary y category dejando así la opción de introducir información adicional que no siga una estructura fija en mongo (por ejemplo si se incluye una foto en la review)

SEGUNDA PARTE: MENÚ DE VISUALIZACIÓN

Esta parte ha sido probablemente una de las que más desafíos ha presentado de todo el proyecto. Para abordarla comenzamos con estructurar las queries necesarias para la visualización de los datos y las probamos en ambas plataformas en función de la información que quisiéramos obtener. A continuación comenzamos con el desarrollo del menú interactivo, comenzamos con dashboard, plataforma que utilizamos desde python el cuatrimestre pasado y con la que ya estábamos familiarizados. El problema que nos encontramos aquí fue una limitación en cuanto al diseño que teníamos en mente, llevándonos a explorar otras opciones. En esta búsqueda de soluciones encontramos streamlit, una librería de python muy interesante con la que habíamos desarrollado un proyecto personal y parecía cumplir nuestros requisitos. Por tanto fuimos construyendo de manera gradual nuestra interfaz gráfica, añadiendo funcionalidades y algunas personalizaciones por el camino (véase por ejemplo añadir una máscara al wordcloud en función de la categoría).

- El comando para ejecutar el menú es “streamlit run <ruta_archivo.py>”

TERCERA PARTE: APLICACIÓN PYTHON + NEO4J

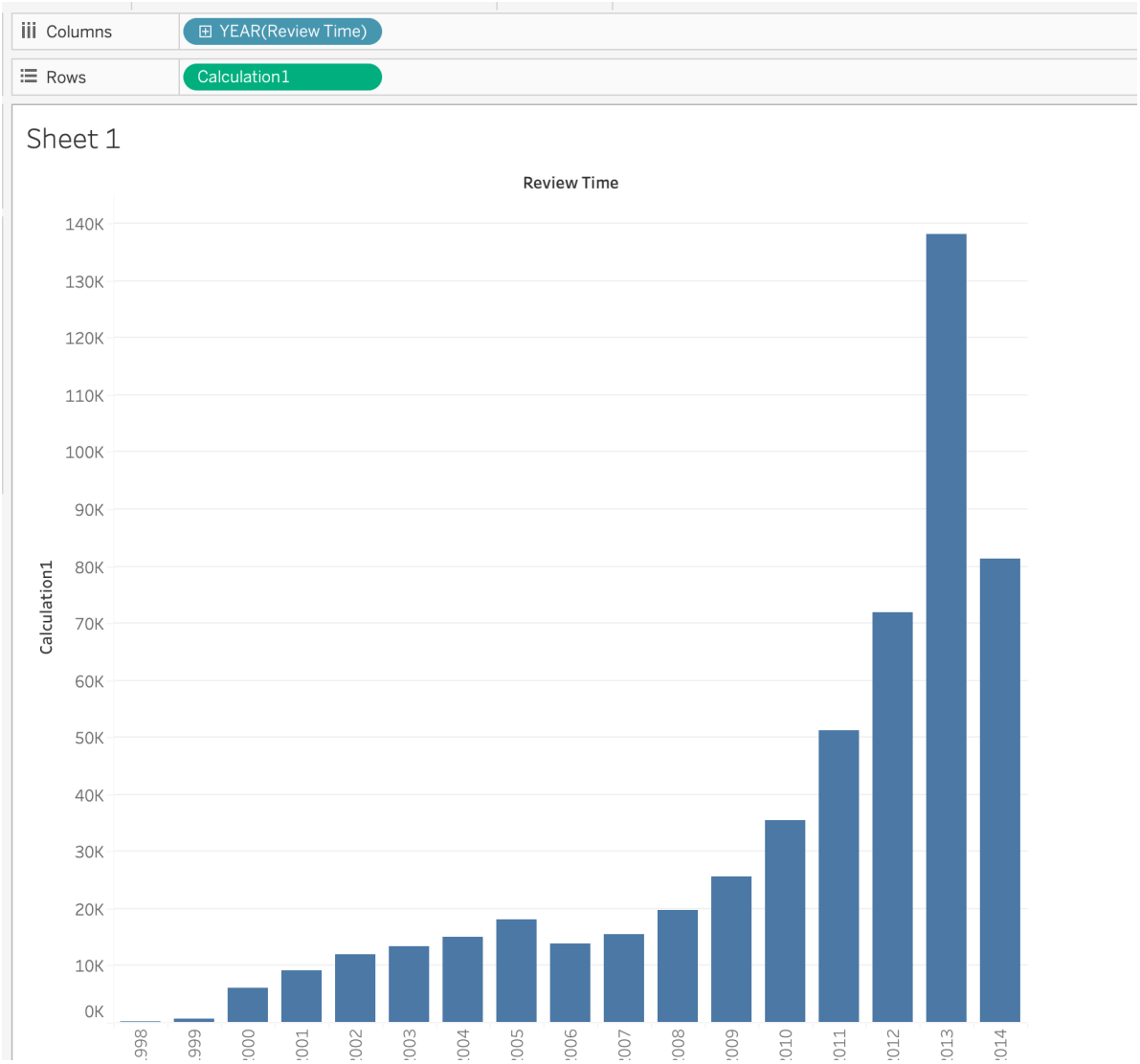
Al ser una de las últimas bases de datos con las que habíamos trabajado neo4j resultó relativamente fácil en comparación con otras partes del proyecto. Aun así, cabe destacar que durante el apartado 1, tratamos de obtener directamente las intersecciones y uniones entre usuarios a través de sql y esto resultaba muy lento computacionalmente, por lo que decidimos hacerlo mediante sets de python, reduciendo el número de consultas.

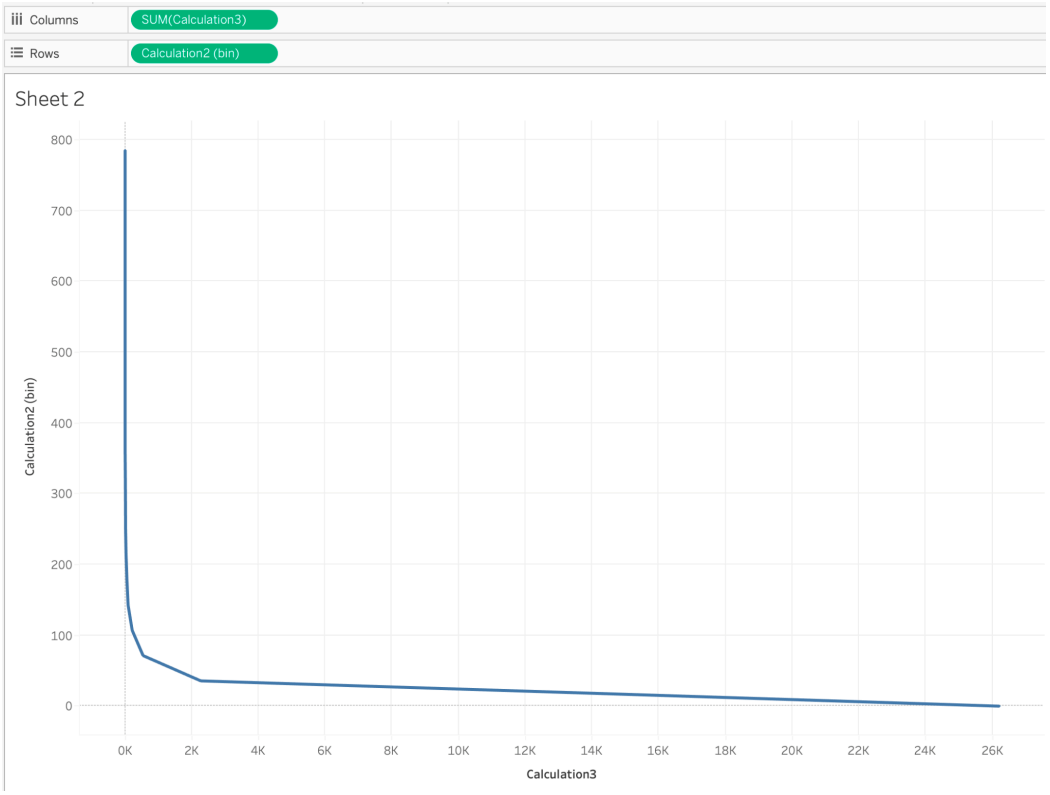
CUARTA PARTE: NUEVOS DATOS

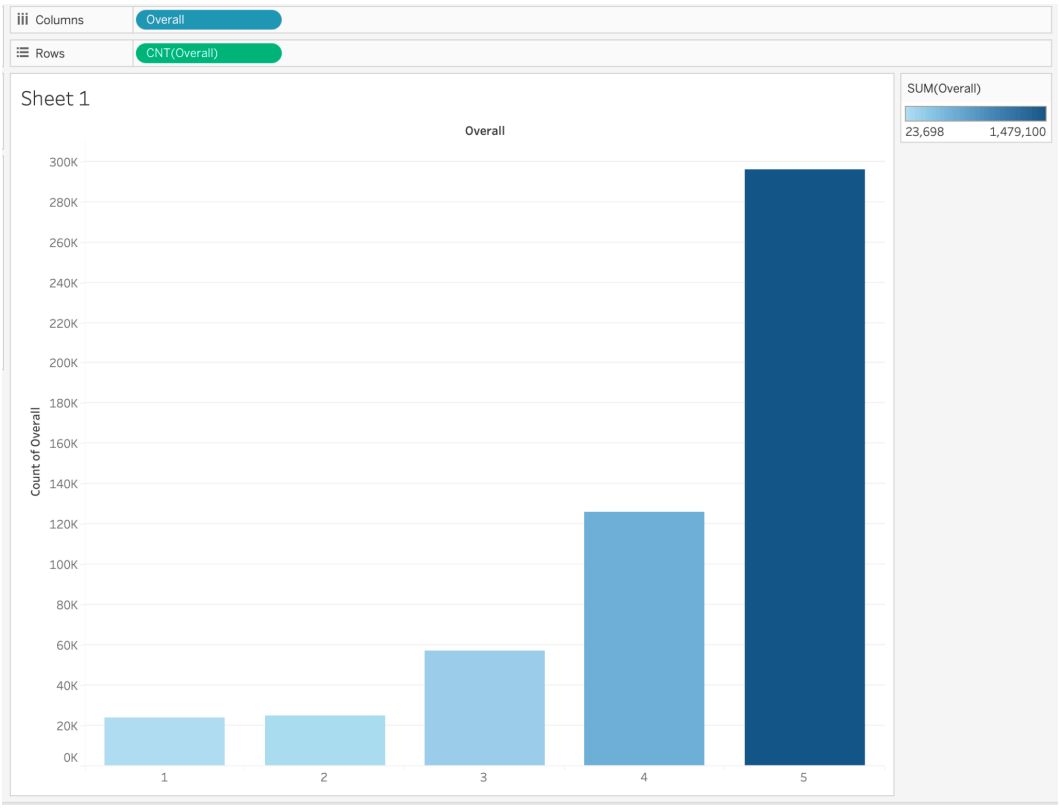
Para esta parte tomamos como plantilla nuestro fichero de carga de datos inicial, con el ligero matiz de que hemos creado otro archivo de configuración asociado a este nuevo fichero de carga de datos, en el cual la diferencia más notable es que el id propio que habíamos introducido para cada review ahora se obtiene mediante una consulta, siendo el último id += 1. Cabe recalcar en este punto que hay un segundo archivo de configuración llamado configuracion_nuevos_datos.py que será el empleado en este apartado.

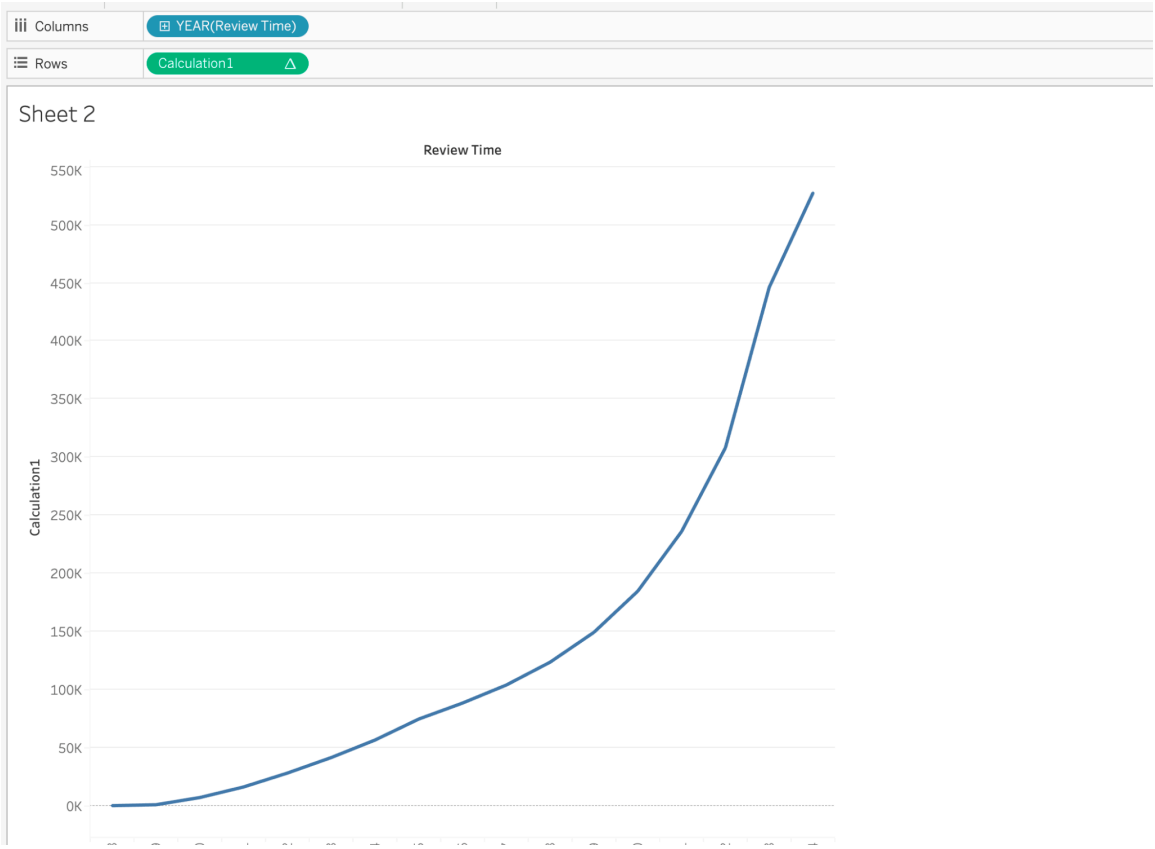
QUINTA PARTE: MÁS VISUALIZACIÓN Y MACHINE LEARNING

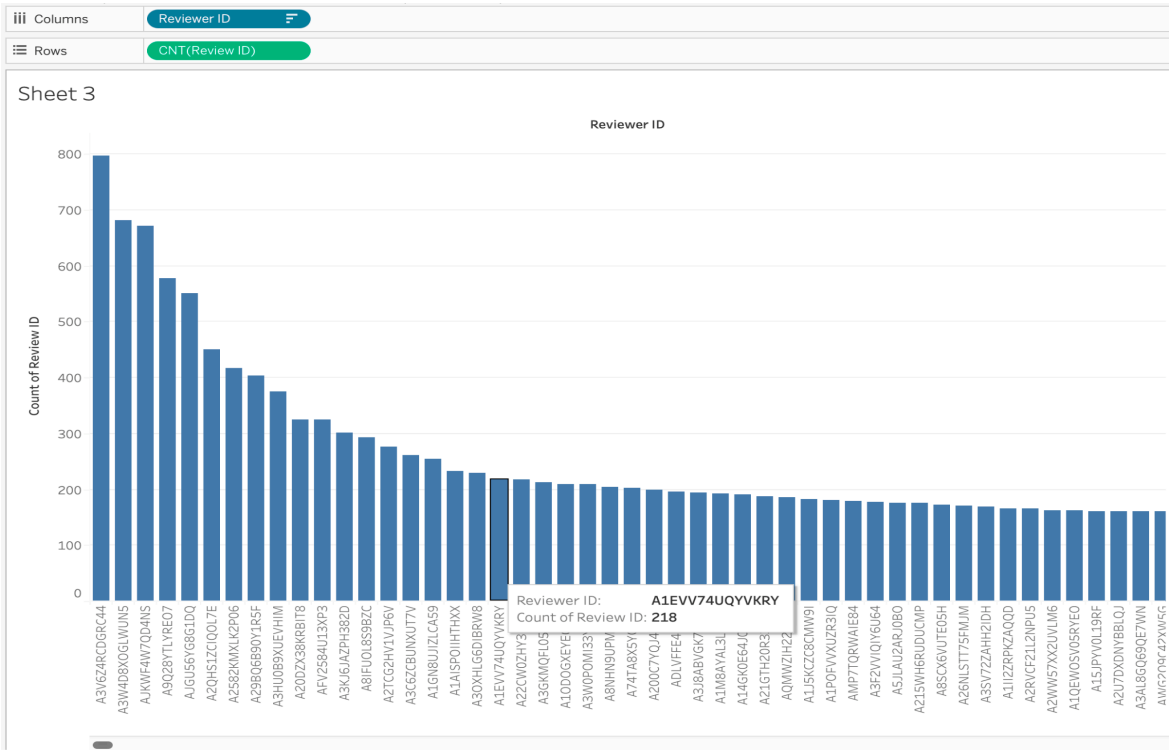
Aquí está la visualización mostrada en nuestro menú del apartado 2, pero ahora realizado mediante Tableau:




Columns
SUM(Calculation3)
Rows
Calculation2 (bin)
Sheet 2
800
700
600
500
400
300
200
100
0
Calculation2 (bin)
0K
2K
4K
6K
8K
10K
12K
14K
16K
18K
20K
22K
24K
26K
Calculation3


Columns
Overall
Rows
CNT(Overall)
Sheet 1
Overall
SUM(Overall)
23,698
1,479,100
300K
280K
260K
240K
220K
200K
180K
160K
140K
120K
100K
80K
60K
40K
20K
0K
Count of Overall
1
2
3
4
5


Columns
YEAR(Review Time)
Rows
Calculation1
Sheet 2
Review Time
Calculation1
550K
500K
450K
400K
350K
300K
250K
200K
150K
100K
50K
0K


Columns
Reviewer ID
Rows
CNT(Review ID)
Sheet 3
Reviewer ID
Count of Review ID
800
700
600
500
400
300
200
100
0
Reviewer ID: A1EVV74UQYVKRY
Count of Review ID: 218

Respecto al modelo de machine learning sugerido, tras barajar todas las opciones vistas este curso, queríamos algo que fuera realista de aplicar, a la vez de adaptarse a los datos de entrada disponible y diese sugerencias coherentes. Por ello hemos optado por un modelo de K-Nearest Neighbours (KNN), el cual toma los k vecinos más cercanos al usuario a evaluar y devuelve el artículo más popular que no haya comprado el usuario a recomendar pero si la mayor parte de los k vecinos. La vuelta de tuerca de este algoritmo y por lo que nos parece que puede ser una gran opción es debido a que esos k vecinos se obtienen mediante la similitud de Jaccard, por lo que será una recomendación basada en lo que le gusta a perfiles muy similares.