

# PROYECTO FINAL BASES DE DATOS

Andrés Gil Vicente  
Jorge Carnicero Príncipe

22/04/2025

# Índice

|  |           |
|--|-----------|
| <b>1. Información y datos del proyecto</b>   | <b>3</b>  |
| <b>2. Proceso de diseño y carga de datos</b>   | <b>3</b>  |
| 2.1. Creación de índices adicionales . . . . .   | 4         |
| 2.2. Fichero load_data.py . . . . .  | 4         |
| 2.3. Fichero configuracion.py . . . . .  | 5         |
| 2.4. Esquema Relacional MySQL . . . . .  | 6         |
| 2.5. Colección MongoDB . . . . .   | 6         |
| <b>3. Aplicación Python de visualización</b>   | <b>6</b>  |
| 3.1. Evolución de reviews por años. . . . .  | 7         |
| 3.2. Evolución de la popularidad de los artículos. . . . .   | 7         |
| 3.3. Histograma por nota. . . . .  | 8         |
| 3.4. Evolución de las reviews a lo largo del tiempo . . . . .  | 9         |
| 3.5. Histograma de reviews por usuario . . . . .   | 9         |
| 3.6. Obtener nubes de palabras . . . . .   | 10        |
| 3.7. Gráfica distinta - Pie Plot con la media de caracteres en función del overall y la categoría del producto . . . . . | 10        |
| <b>4. Aplicación Python y Neo4J</b>  | <b>11</b> |
| 4.1. Obtener similitudes entre usuarios y mostrar enlaces en Neo4J . . . . .   | 11        |
| 4.2. Obtener enlaces entre usuarios y artículos . . . . .  | 12        |
| 4.3. Obtener algunos usuarios que han visto más de un determinado tipo de artículo . . . . .                             | 13        |
| 4.4. Artículos populares y artículos en común entre usuarios . . . . .   | 13        |
| <b>5. Inserción de ficheros adicionales</b>  | <b>14</b> |
| <b>6. Aplicación Machine Learning</b>  | <b>14</b> |
| 6.1. Representación gráficos en Tableau . . . . .  | 14        |
| 6.2. Implementación Machine Learning para recomendación . . . . .  | 17        |
| <b>7. Opcional: Implementación modelo ML</b>   | <b>17</b> |
| <b>8. Anexo: Imágenes Adicionales</b>  | <b>19</b> |

## 1. Información y datos del proyecto

- **Autores del proyecto:** Andrés Gil Vicente y Jorge Carnicero Príncipe.
- **Curso:** 2º A iMAT
- **Asignatura:** Bases de Datos
- **Fecha de entrega:** 22/04/2025

## 2. Proceso de diseño y carga de datos

El primer paso del proyecto es determinar cuál va a ser el diseño de nuestras bases de datos, en las cuales vamos a almacenar inicialmente la información procedente de los siguientes ficheros:

- Toys and Games 5-core (Toys\_and\_Games\_5.json, 167,597 reviews).
- Video Games 5-core (Video\_Games\_5.json, 231,780 reviews).
- Digital Music 5-core (Digital\_Music\_5.json, 64,706 reviews)
- Musical Instruments 5-core (Musical\_Instruments\_5.json, 10,261 reviews)

Es importante tener en cuenta que la estructura de cada una de las filas de los ficheros json viene ordenada mediante el siguiente esquema:

```
- reviewerID: "A2HD75EMZR8QLN" # Es el autor de la review
- asin: "0700099867" # ID del artículo
- reviewerName: "123" # Nombre del reviewer
- helpful: [8, 12] # Array conteniendo la utilidad
- reviewText: "texto" # Texto de la review
- overall: 5 # Nota
- summary: "texto" # Resumen de la review
- unixReviewTime: 1341792000 # Timestamp
- reviewTime: "07 9, 2012" # Fecha de la review
```

Analizando el tipo de dato que se almacena en cada uno de los campos anteriormente nombrados, hemos tomado la decisión de distribuir los datos en nuestras bases de datos en función de si son estructurados o no. Este ha sido el criterio seguido para diseñar la estructura global del proyecto y de sus distintas bases de datos. Si los datos son estructurados, los almacenaremos en la base de datos de MySQL, mientras que si los datos no son estructurados, entonces los almacenaremos en la base de datos de MongoDB.

De este modo, vemos que los campos reviewerID, asin, reviewerName, overall, unixReviewTime y reviewTime contienen **datos estructurados**, por lo que los almacenaremos en tablas de MySQL. En cambio, los campos de helpful, reviewText y summary contienen datos de texto largo o de tipo array, por lo que son **datos no estructurados** y por ello decidimos guardarlos en una colección de MongoDB.

## 2.1. Creación de índices adicionales

Si exploramos los datos, vemos que tanto el identificador de reviewer (reviewerID), como el identificador de producto (asin) tienen un formato poco amigable, con el cuál sería más incómodo trabajar. Por ese motivo y por facilitar el manejo de los datos, para cada entidad hemos decidido crear nosotros mismos un **identificador numérico adicional** al que ya tenían. De esta forma podremos trabajar con los datos de manera más sencilla y clara. Por ejemplo, usaremos identificadores del tipo “1” en vez de otros como “5555991584”, aunque guardaremos ambos.

Las entidades para las que vamos a crear identificadores adicionales son las reviews, las personas que hacen reviews, los productos y los tipos de productos; no obstante, esto se detallará a continuación cuando mostremos el esquema relacional de las tablas de SQL y también la estructura de la colección de MongoDB.

## 2.2. Fichero load\_data.py

Este script se empleará para **cargar toda la infraestructura inicial del proyecto** a partir de los distintos ficheros de datos, creando así distintas bases de datos, tanto de MongoDB como de SQL. Estas tendrán sus respectivas tablas y colecciones. Las variables con los nombres de las rutas de los ficheros, nombres de las bases, credenciales para hacer las conexiones y demás detalles, se importarán desde el fichero de configuracion.py .

Además, cabe mencionar que cada vez que se ejecuta el script, borramos todas las bases de datos en caso de que ya existan, y una vez se ha hecho eso, ya se vuelven a cargar por completo. Esto se hace para **evitar posibles errores de duplicación de datos, inconsistencias o incoherencias**. Es más, este planteamiento facilita la ejecución sucesiva del programa, sin necesidad de borrar manualmente las bases de datos desde el Workbench de SQL, desde el MongoDB Compass o desde la propia shell.

Otra de las optimizaciones que hemos realizado para acelerar el proceso de cargado de datos y que no sea tan costoso, es plantear una **inserción por lotes de los datos**. Para ello, inicialmente hemos fijado un tamaño de lote, el cual hemos almacenado en una variable global en configuracion.py para que sea fácil y cómodo de cambiar en caso de que se necesite. El funcionamiento de la inserción por lotes consiste en ir leyendo los ficheros línea a línea y extraer los datos que necesitamos para construir las tuplas (SQL) o diccionarios (MongoDB) que luego vamos a insertar en las tablas o colecciones correspondientes.

En cada una de esas lecturas de una línea, en vez de hacer directamente la inserción, guardaremos los datos en listas y una vez el tamaño de dichas listas alcance el tamaño umbral, dado por el tamaño del lote, previamente establecido, entonces ya sí que haremos una inserción múltiple con todos los elementos de la lista, y volveremos a comenzar el proceso vaciando dicha lista. De esta forma, **no saturaremos el sistema haciendo un número excesivo de inserciones**, sino que solo haremos una inserción múltiple cada cierto tiempo. Si el proceso de inserción termina y las listas de los lotes aún contienen datos, insertaremos todos los datos que queden. Esta estructura se ha empleado tanto para SQL como para MongoDB.

Cabe destacar también, que la inserción de datos de nuestro programa **tiene en cuenta la posibilidad de existencia de valores repetidos** (por ejemplo la misma persona puede hacer varias reviews o del mismo producto se pueden hacer reviews distintas), por lo que cuando se va a introducir un dato en las tablas de SQL, en caso de que ya exista dicha información, lo que se hace es actualizar los datos referentes a esa entidad, por si acaso hay información relevante que se deba almacenar en la base de datos. Eso sí, en caso de que la información que vaya a ser introducida en sustitución de la que ya existía, sea un dato nulo, entonces no se insertará para tratar de preservar la máxima información posible en el sistema. Todo esto se gestiona con este procedimiento tanto en load\_data.py como en inserta\_dataset.py, el cual detallaremos más adelante.

Solo como comentario, queremos decir que hemos probado a crear **índices en la base de datos de SQL**, para que las posteriores consultas fueran más eficientes y rápidas; no obstante, al comparar el rendimiento de dichas consultas con índices y sin índices, hemos decidido no implementarlos ya que las mejoras no son apreciables, al menos con esta cantidad de datos.

Por último, a nivel de tratado de datos y su formato, hemos decidido **convertir los datos del campo de reviewTime a formato date con la estructura (YYYY-MM-DD)**. Gracias a esta conversión, pasaremos por ejemplo de formatos de tipo “07 9, 2012”, a formatos de tipo “2012-07-09”. Esto permite tener la información de estas fechas más controlada y trabajar más cómodamente con ella.

### 2.3. Fichero configuracion.py

Este script se emplea para **almacenar las variables globales** que contienen las rutas de los ficheros de datos, así como los nombres de las distintas bases de datos, tanto de SQL como de MongoDB, las distintas tablas y colecciones y las credenciales necesarias para poder hacer las conexiones correctamente.

Si se desea desplegar el proyecto y se tienen almacenados los ficheros en otra carpeta distinta, o se dispone de otro usuario o contraseña para hacer alguna de las conexiones, entonces será suficiente con modificar las variables desde este fichero y ya se podrán ejecutar el resto de archivos con normalidad. En este fichero se decidirán también qué archivos se van a cargar en las bases de datos empleando load\_data.py o inserta\_dataset.py, así como otras variables de suma importancia, como por ejemplo el tamaño de los lotes de inserción.

## 2.4. Esquema Relacional MySQL

A continuación mostramos el esquema relacional de las tablas que hemos creado en nuestra base de datos de SQL. Recordemos que solo contiene datos de tipo estructurado ya que el resto de datos se almacenarán en MongoDB.

- **Personas** (id\_persona, reviewerID, reviewerName)
- **Tipos\_producto** (tipo\_producto, nombre\_tipo\_producto)
- **Productos** (id\_producto, asin, tipo\_producto)
  - *tipo\_producto* FK REFERENCES **Tipos\_producto**(tipo\_producto)
- **Review** (id\_review, id\_persona, id\_producto, overall, unixReviewTime, reviewTime)
  - *id\_persona* FK REFERENCES **Personas**(id\_persona)
  - *id\_producto* FK REFERENCES **Productos**(id\_producto)

## 2.5. Colección MongoDB

Por otro lado, tenemos que el formato a priori de los documentos de la colección de reviews en MongoDB será de la siguiente forma:

- **Review** (id, helpful, reviewText, summary)
  - *\_id* es el identificador único de cada documento. Lo hemos creado nosotros de forma adicional para llevar un mayor control de los datos. Este identificador coincide con el de las reseñas de la tabla de Review de SQL.
  - *helpful* es una lista o array de dos enteros.
  - *reviewText* contiene el texto completo de la reseña (tipo **string** largo).
  - *summary* contiene un resumen corto de la reseña (tipo **string**).

## 3. Aplicación Python de visualización

En este apartado se va a desplegar una aplicación Python que permita a los usuarios el **acceso y visualización de los datos** en función de lo que deseen consultar. Se ofrecerá una interfaz al usuario para que decida las funcionalidades que desea utilizar, mostrándose así un gráfico referente a la información que este haya querido buscar en la base de datos.

Además, se mostrarán **mensajes indicativos con distintos colores** para facilitar la comprensión de los usuarios y hacer que su experiencia sea más amena. En caso de cualquier error, se recogerán las excepciones para que el programa no termine de forma abrupta en ningún caso.

### 3.1. Evolución de reviews por años.

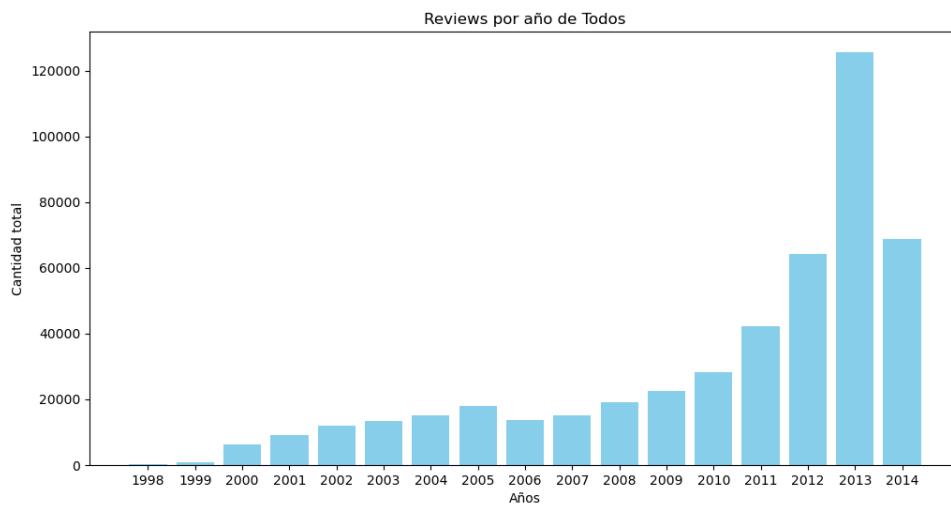


Figura 1: Evolución de reviews por año, todas las categorías juntas.

### 3.2. Evolución de la popularidad de los artículos.

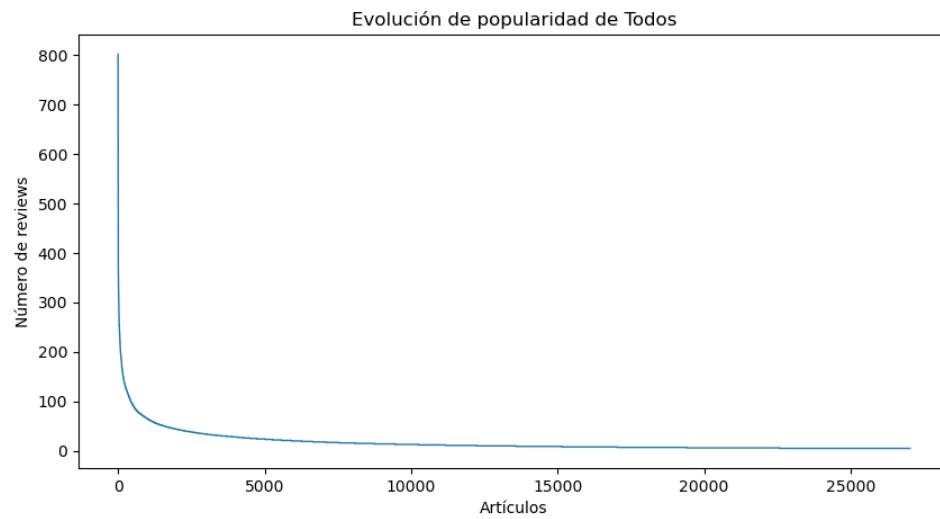


Figura 2: Evolución de popularidad de los artículos en todas las categorías juntas.

### 3.3. Histograma por nota.

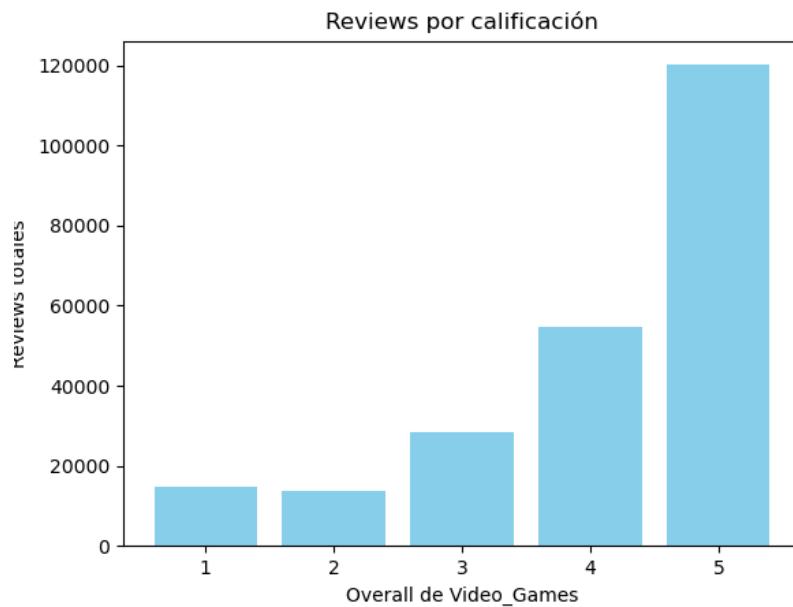


Figura 3: Histograma por nota de Videogames.



Figura 4: Histograma por nota de un producto concreto (asin:B00000017R)

### 3.4. Evolución de las reviews a lo largo del tiempo

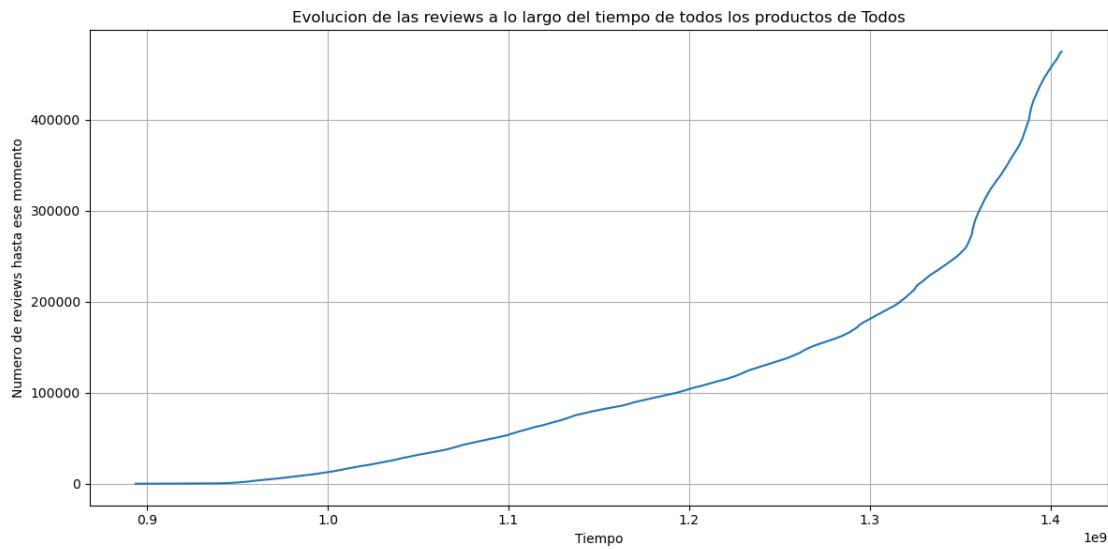


Figura 5: Evolución de las reviews a lo largo del tiempo en todas las categorías juntas

### 3.5. Histograma de reviews por usuario

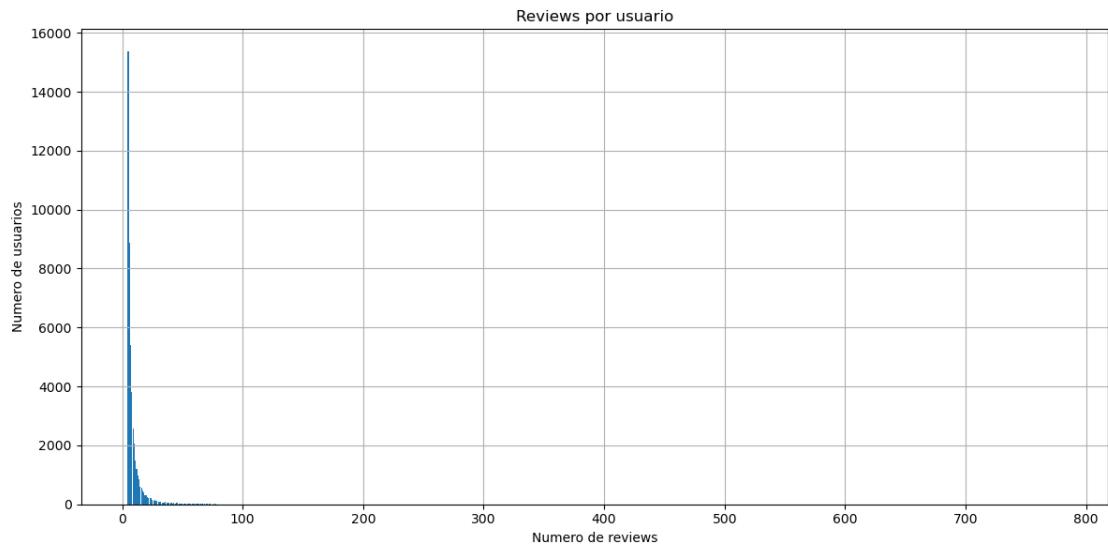


Figura 6: Histograma cantidad de reviews publicadas y cantidad de usuarios

### 3.6. Obtener nubes de palabras



Figura 7: Nube de palabras para la categoría de Video Games

### 3.7. Gráfica distinta - Pie Plot con la media de caracteres en función del overall y la categoría del producto

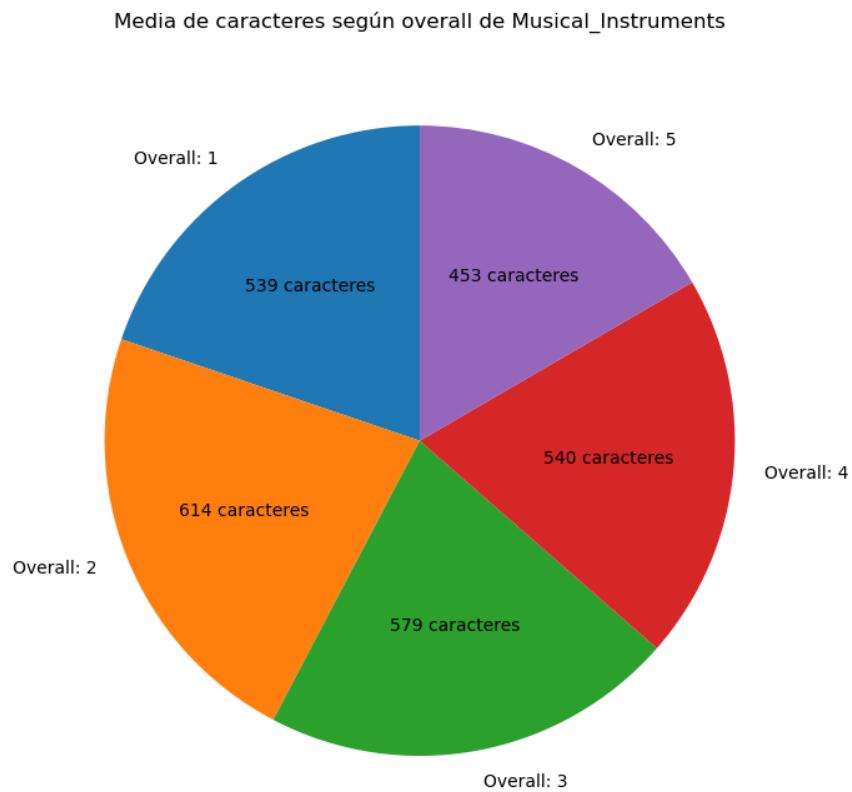


Figura 8: Media de caracteres por overall para la categoría de Musical Instruments

## 4. Aplicación Python y Neo4J

En el fichero Neo4JProyecto.py se ofrecerá una interfaz al usuario para que pueda ejecutar distintas consultas en nuestras bases de datos. En función de los distintos filtros que aplique el usuario en sus búsquedas, **se cargarán los resultados de dicha consulta en Neo4J** y se avisará por pantalla al usuario para que pueda acceder a la base de datos de Neo4J y visualizar dichos resultados. A continuación mostramos algunas de estas imágenes para ilustrar el funcionamiento del programa.

Queremos resaltar que para optimizar la ejecución del programa y que los datos se carguen más rápidamente en Neo4J, empleamos índices sobre los nodos y atributos que corresponde en cada casa. Así, la posterior inserción de datos, que emplea, la sentencia “MERGE”, se vuelve más eficiente.

### 4.1. Obtener similitudes entre usuarios y mostrar enlaces en Neo4J

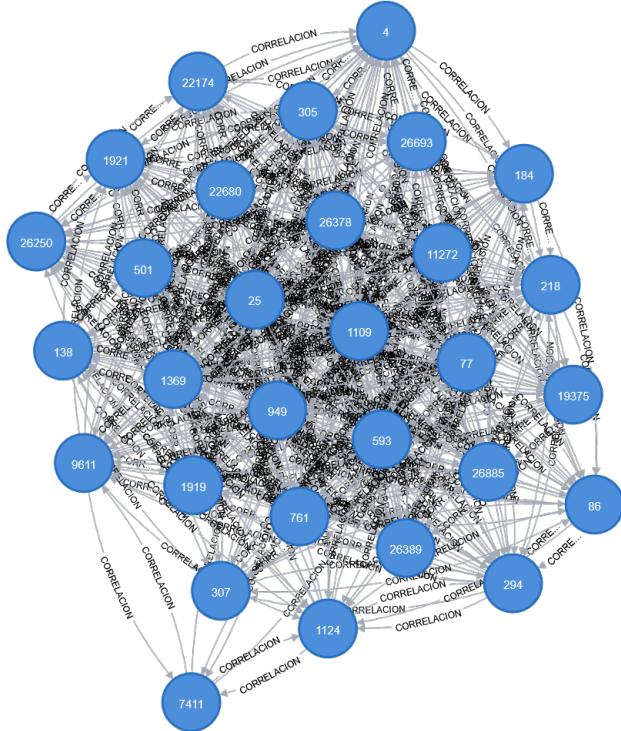


Figura 9: Similitudes entre usuarios

#### 4.2. Obtener enlaces entre usuarios y artículos

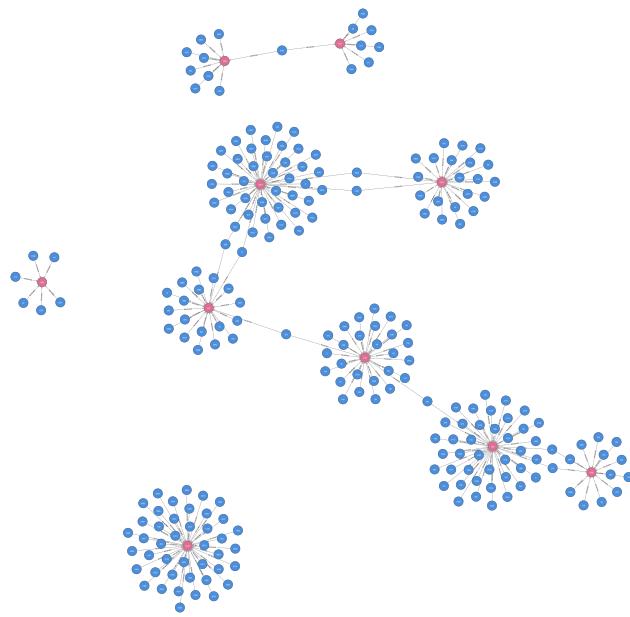


Figura 10: Enlaces entre usuarios y 10 artículos Digital Music

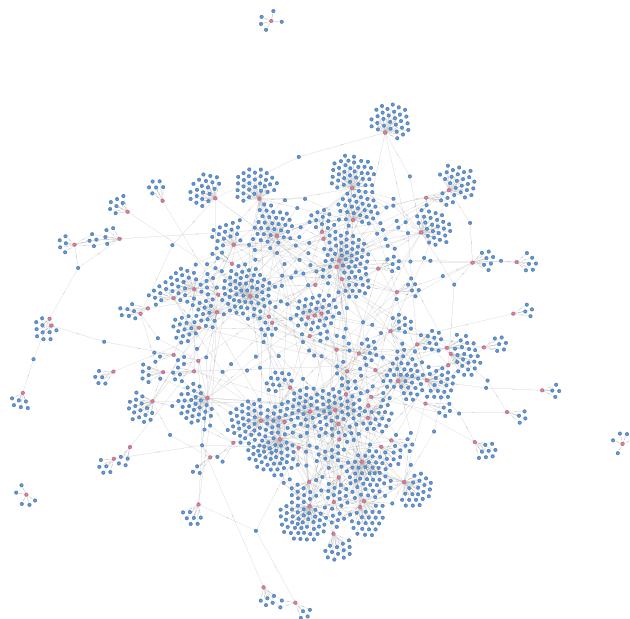


Figura 11: Enlaces entre usuarios y 100 artículos Digital Music

#### 4.3. Obtener algunos usuarios que han visto más de un determinado tipo de artículo

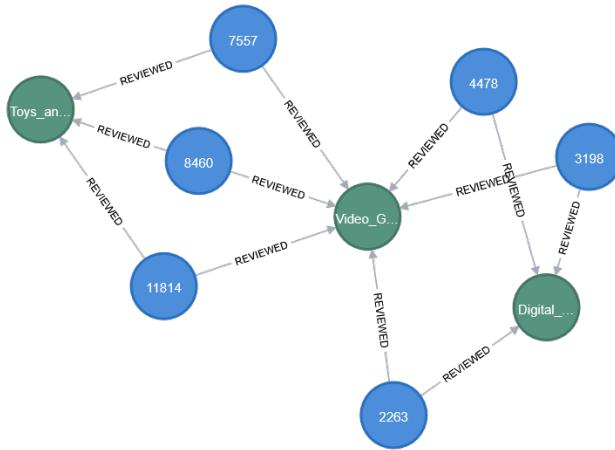


Figura 12: Usuarios con múltiples tipos de artículos

#### 4.4. Artículos populares y artículos en común entre usuarios

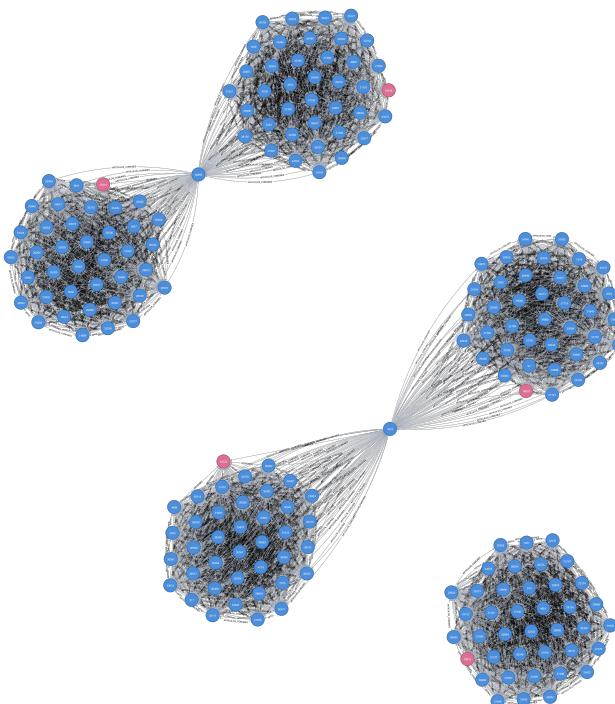


Figura 13: Artículos populares en común

## 5. Inserción de ficheros adicionales

El script `inserta_dataset.py` se empleará para insertar en las bases de datos, uno o más ficheros nuevos, distintos de los que se insertaron en el script de `load_data.py`. El programa está diseñado para que **se puedan insertar tantos ficheros nuevos como se necesite**, si se desea, se puede incluir o dejar de incluir cualquier fichero configurándolo desde las variables globales del archivo `configuracion.py`.

Además, cabe destacar que el procesado de los ficheros se hace línea a línea y que en caso de que un dato ya se haya procesado anteriormente, es decir, que ya existe en la base de datos, entonces se actualizará la información referente a ese dato con la nueva información encontrada. Esto se hará siempre que la nueva información no sea un valor nulo.

De esta forma **gestionamos correctamente la aparición de valores repetidos, sin que haya duplicados incoherentes y permitiendo actualización** de los datos, además de escalabilidad.

## 6. Aplicación Machine Learning

Primeramente, como ya tenemos todos los datos almacenados en nuestra estructura de bases de datos, vamos a mostrar algunas de las gráficas que hemos obtenido anteriormente, pero ahora con una herramienta de visualización más **especializada y profesional, como es Tableau**.

### 6.1. Representación gráficos en Tableau

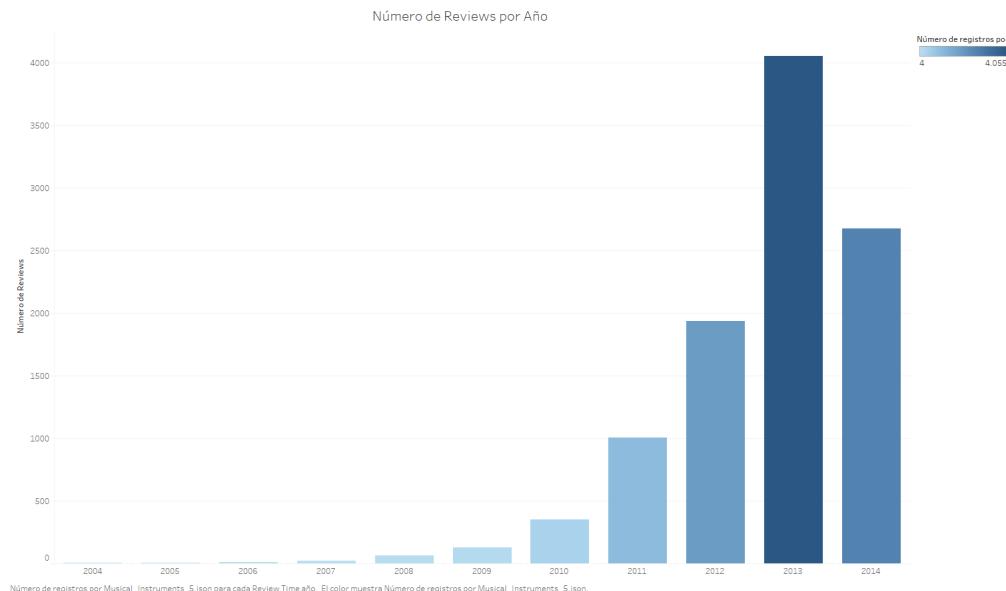


Figura 14: Reviews por año de Musical Instruments (apartado 3.1 Python)

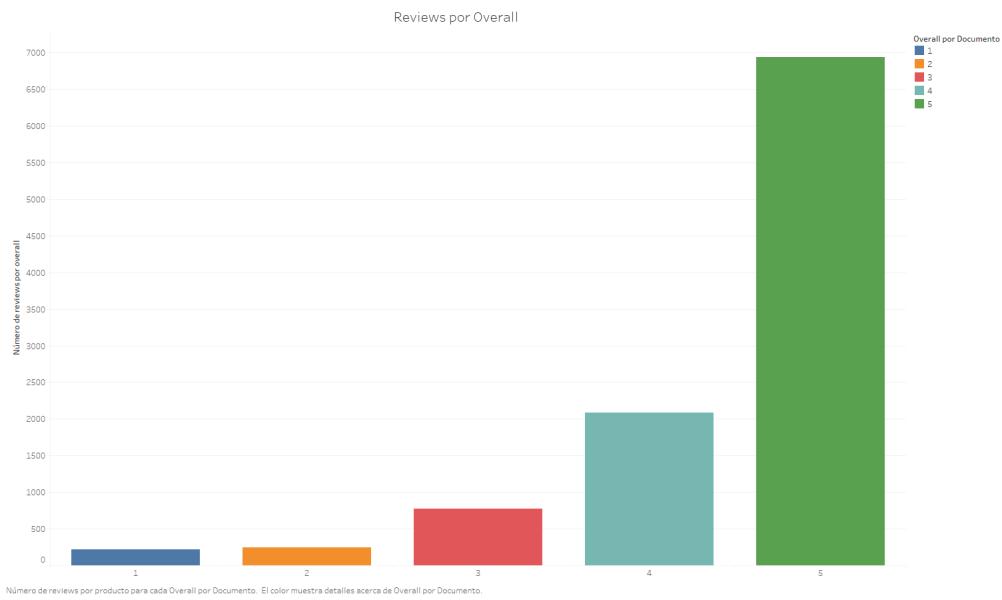


Figura 15: Reviews por overall en Musical Instruments (apartado 3.3 Python)

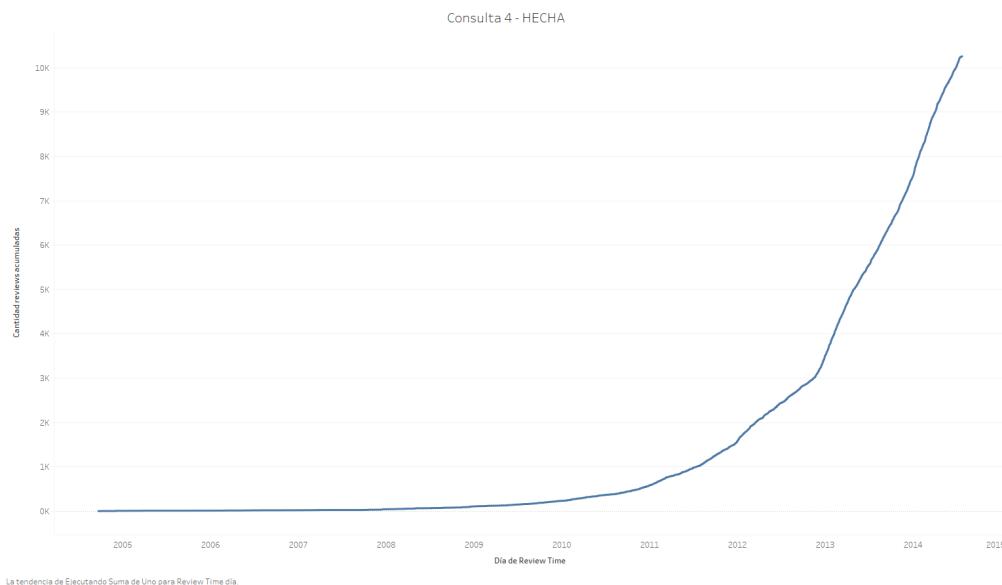


Figura 16: Evolución de reviews en el tiempo en Musical Instruments (apartado 3.4 Python)

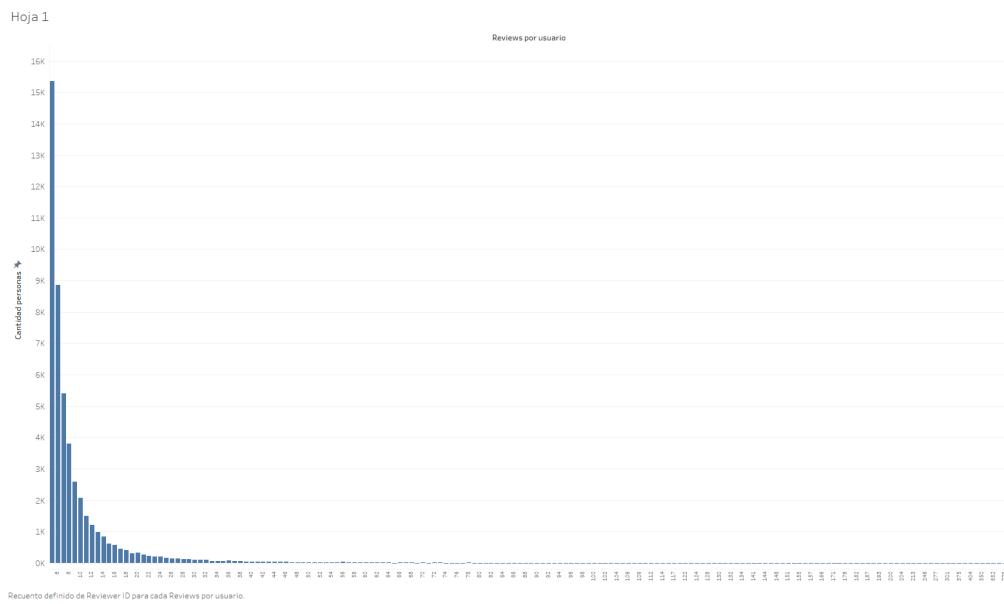


Figura 17: Cantidad de reviews publicadas y cantidad de usuario (apartado 3.5 Python)

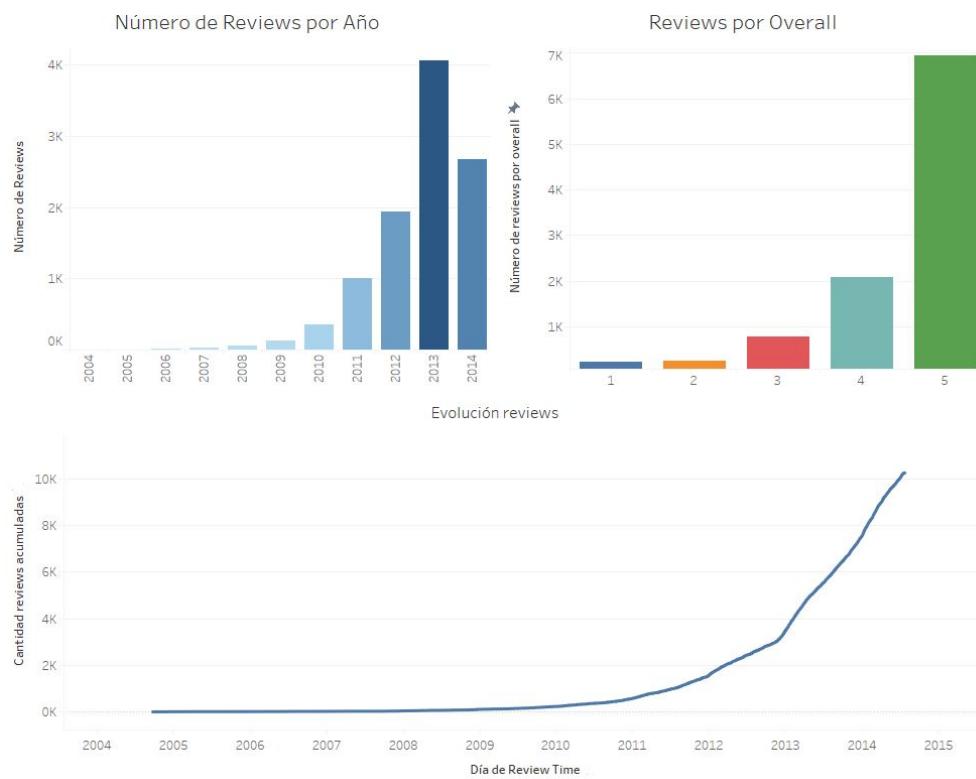


Figura 18: Dashboard en Tableau con gráficos de Musical Instruments

## 6.2. Implementación Machine Learning para recomendación

Nuestra idea de implementación de Machine Learning para la **recomendación de productos a un usuario** se basa en utilizar la similitud entre sus valoraciones y las de otros usuarios. Para ello, calculamos la **correlación de Pearson** entre las valoraciones de nuestro usuario objetivo y las del resto de usuarios, considerando únicamente los productos que estos tienen en común (porque ambos las han reseñado). De esta forma, mediremos hasta qué punto coinciden las puntuaciones (los “overalls”) de nuestros usuarios, y en consecuencia, cómo de afines son a la hora calificar esos producto.

De este modo, si el coeficiente de correlación es alto (por ejemplo, mayor que 0.85), entendemos que ese usuario califica / piensa igual que nuestro usuario al que queremos recomendarle, es decir, que deben compartir criterios similares a la hora de valorar productos. Entonces nos quedaremos con todos los usuarios que tienen una alta correlación y veremos qué productos han valorado estos usuarios, que nuestro usuario objetivo no haya valorado todavía.

Finalmente, entre estos productos candidatos, buscaremos aquellos que hayan recibido valoraciones altas (un overall entre 5 y 4) por parte de los usuarios parecidos. Si hay varios productos y tenemos que desempatar, les daremos **mayor importancia a aquellos productos que hayan sido valorados por un mayor número de usuarios con alta correlación**, ya que estadísticamente,, si les ha gustado a más cantidad de personas de gustos similares, tiene más probabilidad de gustarle también a nuestro usuario.

Este enfoque nos va a permitir aprovechar los patrones de valoración de personas con gustos similares al de nuestro usuario, para realizar recomendaciones personalizadas y proponer **sugerencias sobre productos que aún no ha valorado, porque quizá no conoce todavía.**

## 7. Opcional: Implementación modelo ML

Para llevar a la práctica la idea de la aplicación del modelo de machine learning, **hemos desarrollado el script de machine\_learning.py**, el cual se encarga de plasmar la idea y ejecución de dicha idea. A raíz de los datos que ya teníamos almacenados en nuestras distintas bases de datos, hemos implementado un modelo que se puede emplear para hacer recomendaciones a cualquier usuario sobre cuáles son los productos que seguramente más le pueden interesar y que aún no conoce.

Este proceso se desarrolla analizando los usuarios que tienen gustos similares a los del usuario en cuestión, teniendo en cuenta también si las valoraciones de los productos que tienen en ambos en común son parecidas. De esta forma, podemos sugerir a los usuarios productos nuevos que aún quizás no conocen pero les pueden llegar a interesar.

Está programado de forma que le introduces al programa el id del usuario al cual quieras recomendar, y hará todo el proceso de análisis y recomendación, buscando cuáles son los top 5 productos (top n, fácilmente modificable) que le podemos recomendar a nuestro usuario. Podríamos hacer una recomendación completa de todos los usuarios de la base de datos pero para no saturar la terminal y favorecer la experiencia del usuario, hemos decidido que sea él el que introduzca a qué usuario desea recomendar.

Ejemplos de recomendaciones:

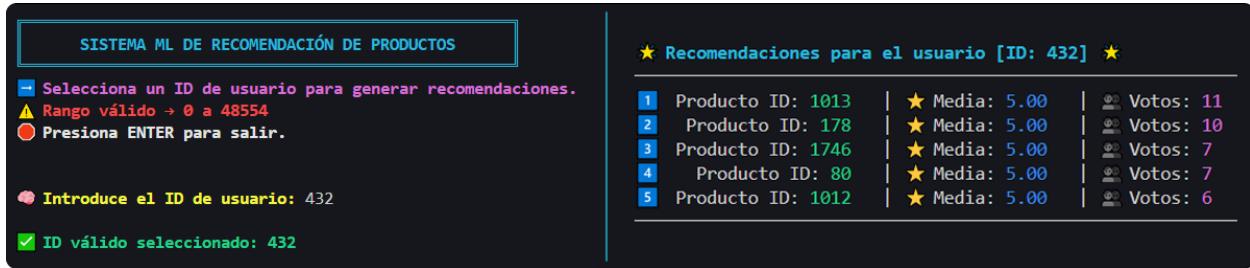


Figura 19: Ejemplo sistema recomendación archivos básicos usuario 432

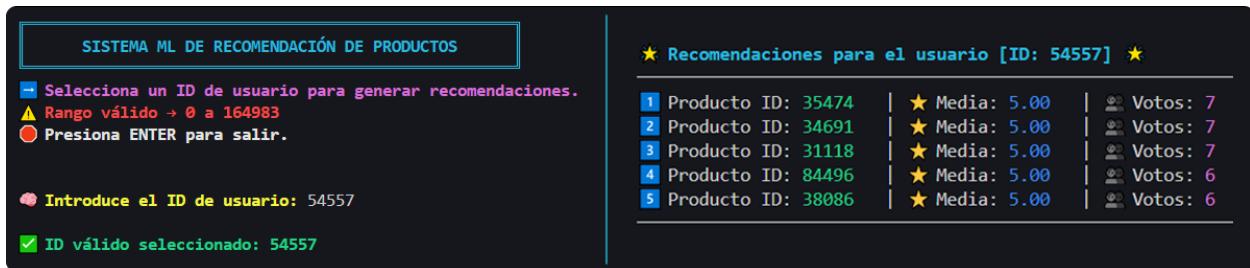


Figura 20: Ejemplo sistema recomendación con nuevos archivos incluidos usuario 54557

Mostramos en forma de interfaz gráfica los resultados de las recomendaciones que se le sugieren al usuario. En estas recomendaciones damos detalles como por ejemplo, la media de puntuaciones que los usuarios afines a él han dado al producto recomendado, o la cantidad de usuarios con gustos parecidos que han recomendado ese producto ya que lo han valorado positivamente.

## 8. Anexo: Imágenes Adicionales

De forma adicional, hemos querido probar a introducir en las bases de datos todos los ficheros disponibles en el repositorio que se mostraba en el enunciado del proyecto. Tras hacer esto, hemos vuelto a generar algunas consultas tanto en SQL como en Noe4J, dando lugar a los siguientes gráficos:

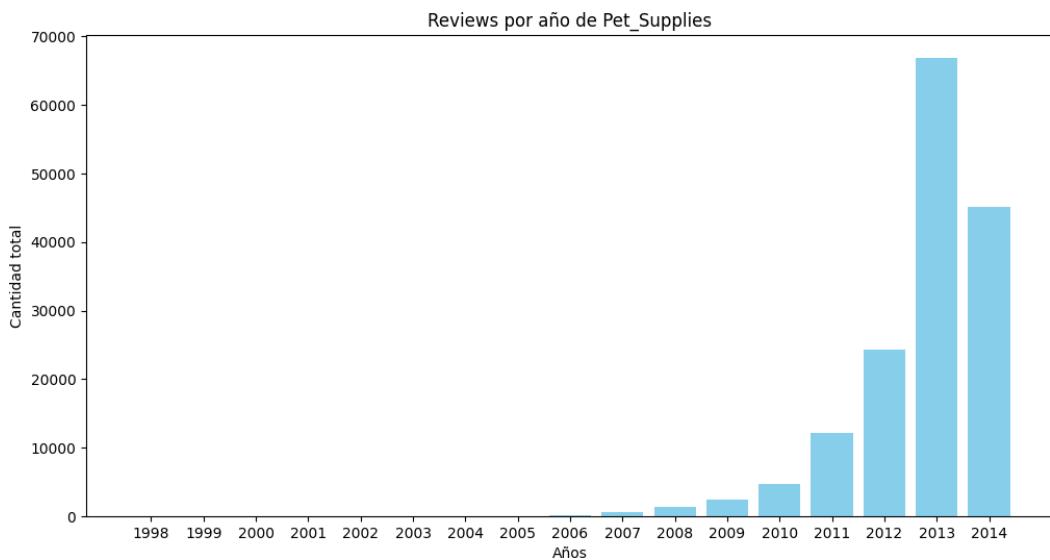


Figura 21: Evolución de reviews por años para Pets and Supplies

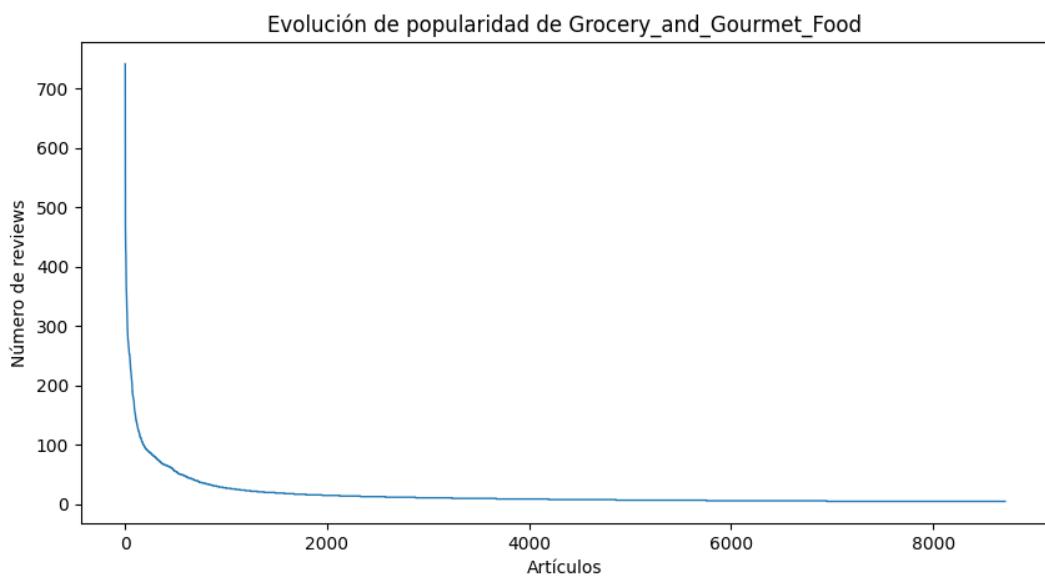


Figura 22: Evolución popularidad por años para Grocery and Gourmet

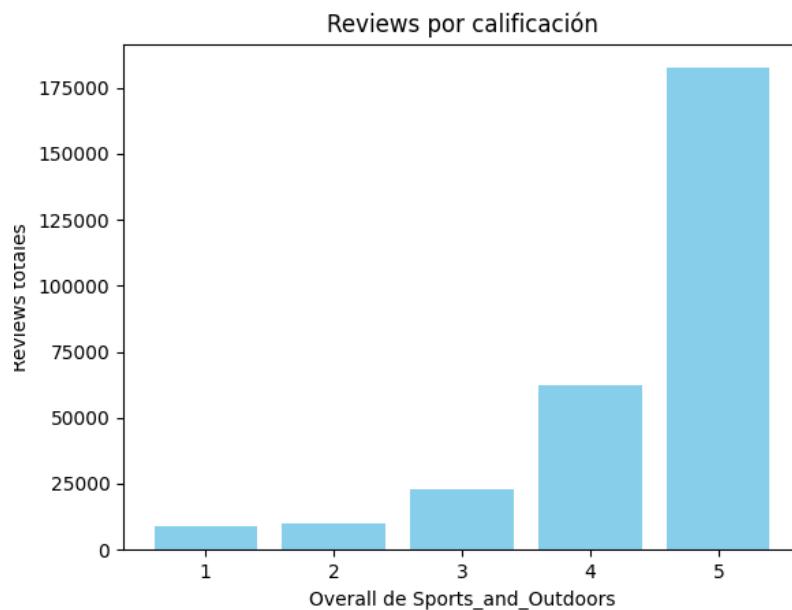


Figura 23: Histograma de reviews por overall en Sports and Outdoor

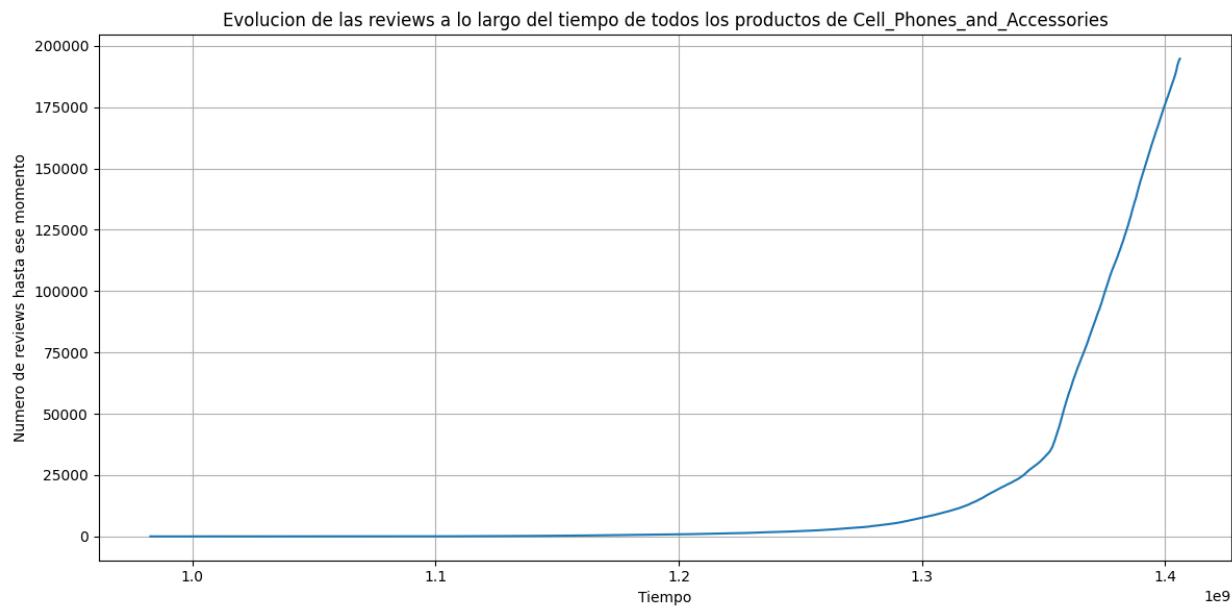


Figura 24: Histograma de reviews por overall en Sports and Outdoor

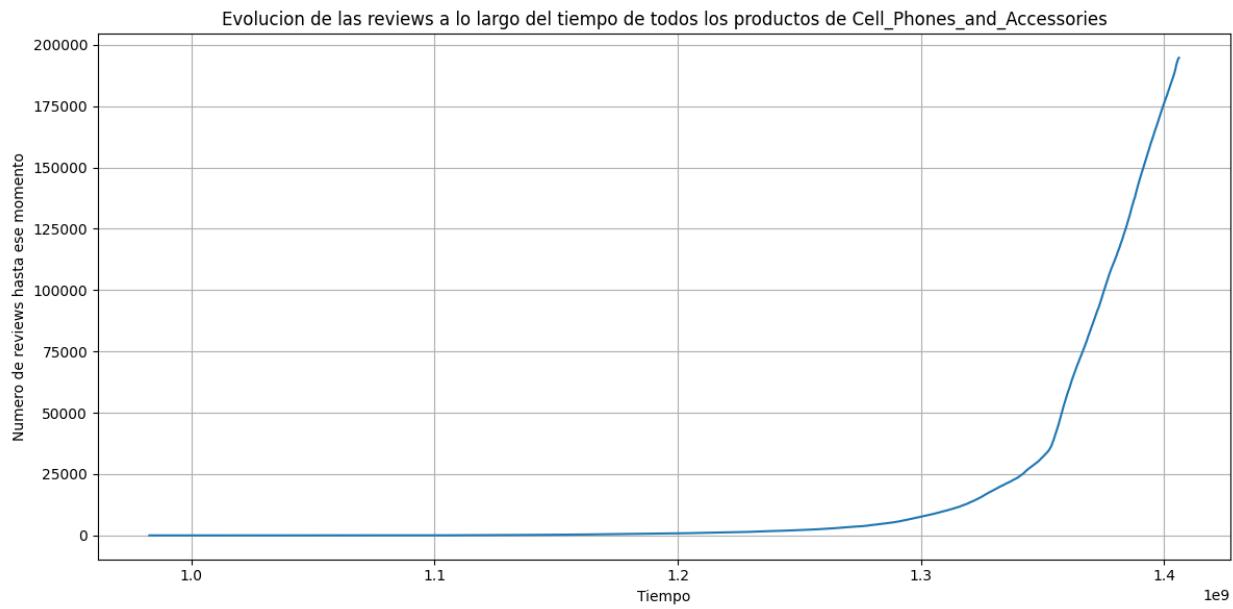


Figura 25: Evolución reviews por años para Cell Phones and Accesories



Figura 26: Nube de palabras para Office Products

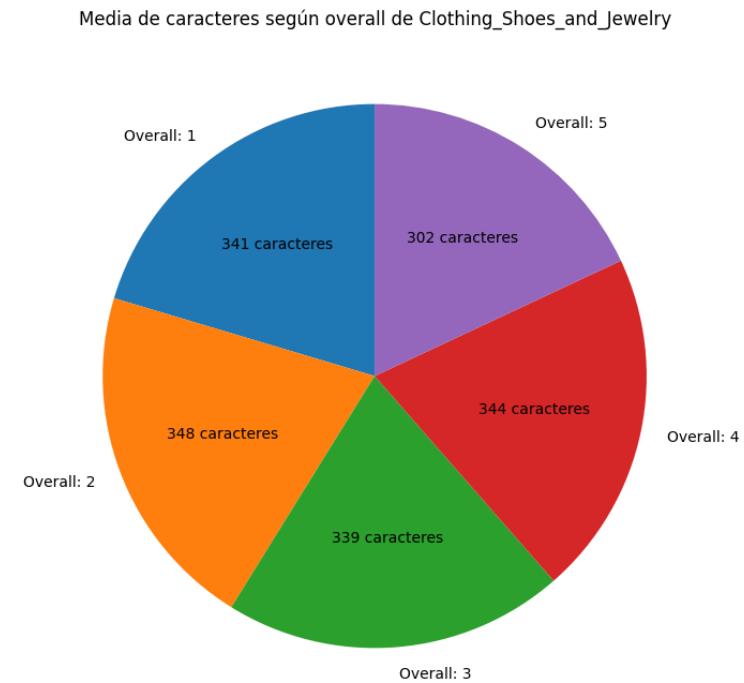


Figura 27: Media de caracteres por overall para Clothing, Shoes and Jewelry

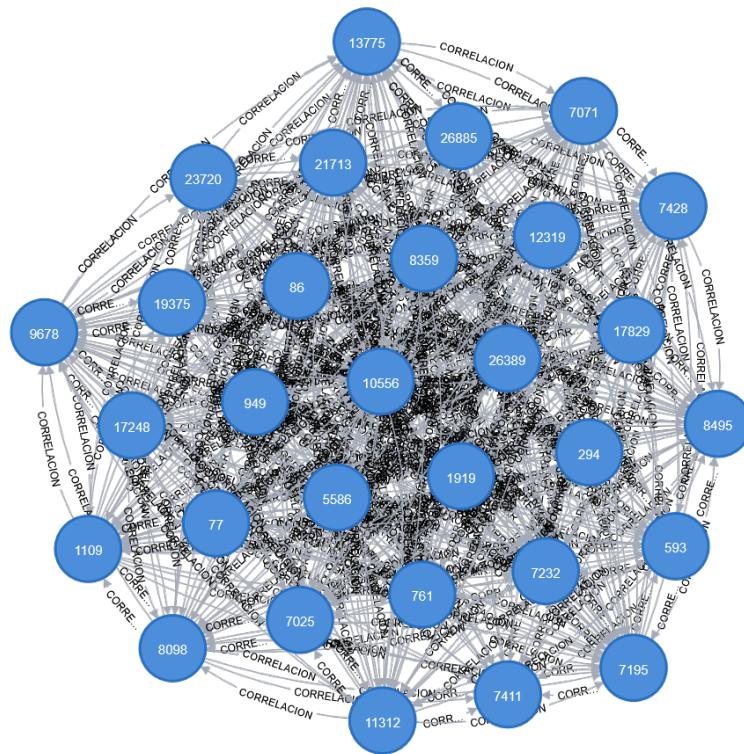


Figura 28: Similitudes entre usuarios - Neo4J

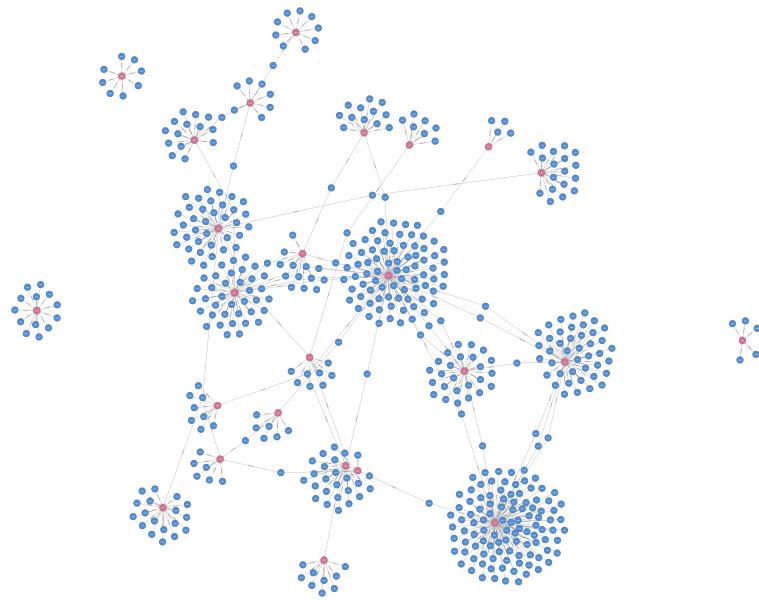


Figura 29: Enlaces entre usuarios y artículos de Amazon Instant Video - Neo4J

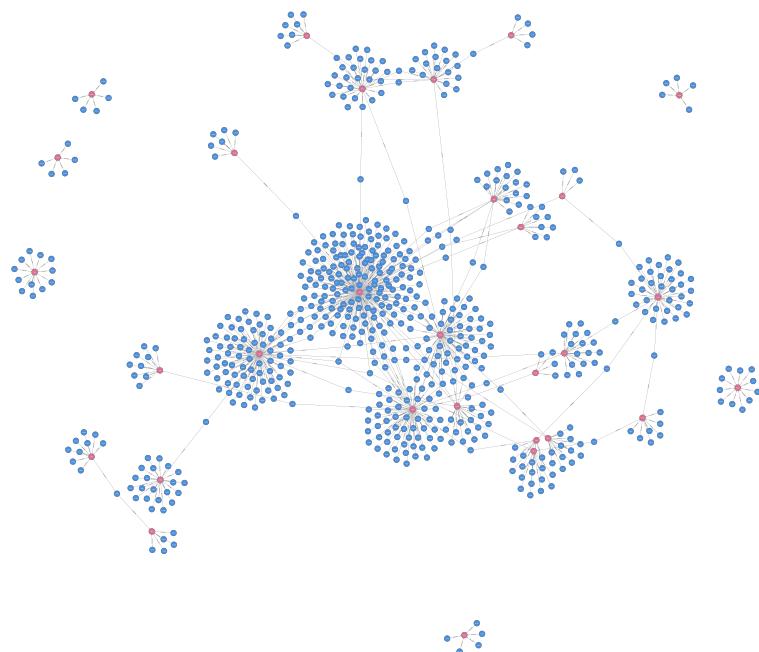


Figura 30: Enlaces entre usuarios y artículos de Office Products - Neo4J

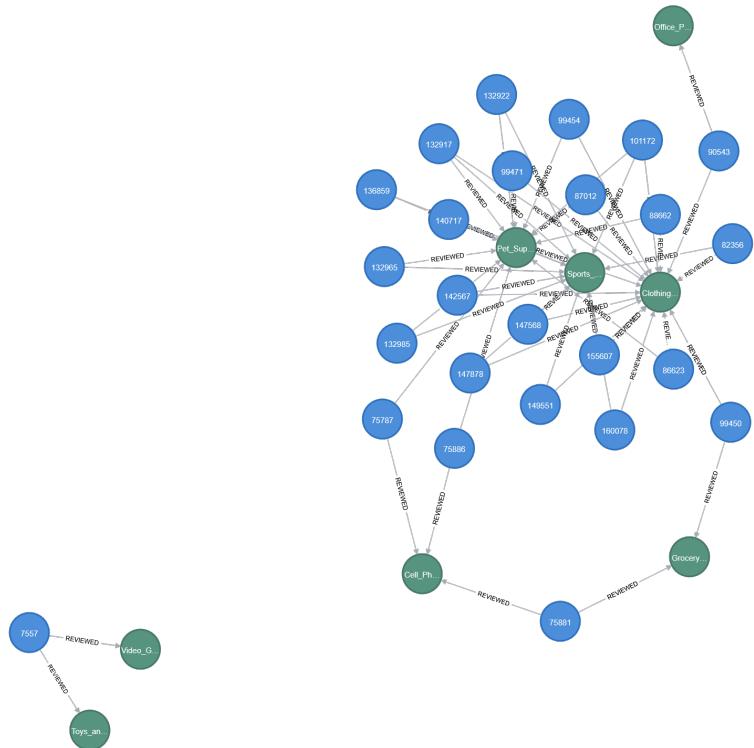


Figura 31: Usuarios con múltiples tipos de artículos - Neo4J

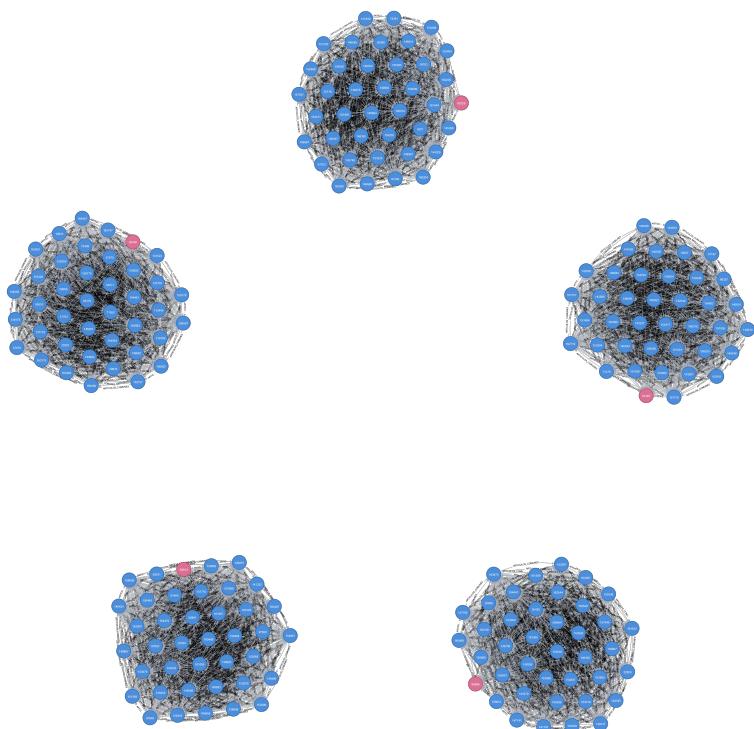


Figura 32: Artículos populares y artículos en común - Neo4J

También nos gustaría mostrar algunas visualizaciones, que por cuestiones de espacio y legibilidad del informe, hemos decidido obviar en apartados anteriores como el de Aplicación Python de Visualización. Por lo tanto, las adjuntamos a continuación:

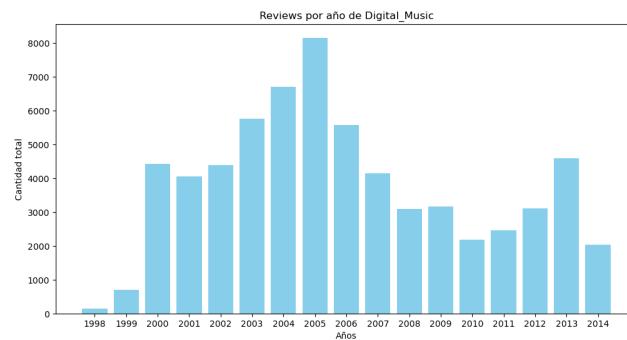


Figura 33: Evolución de reviews por años para Digital Music

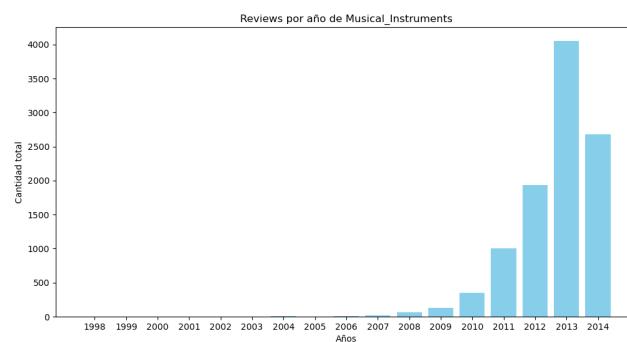


Figura 34: Evolución de reviews por años para Musical Instruments

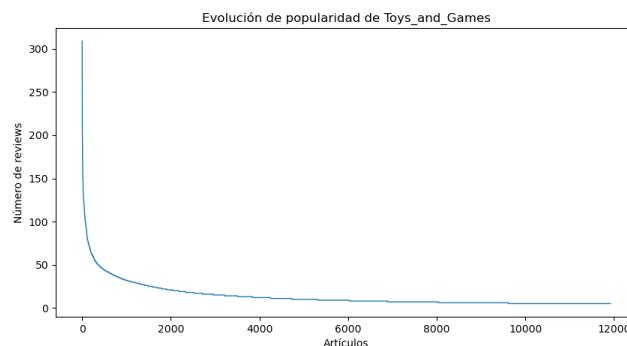


Figura 35: Evolución de la popularidad de Toys and Games

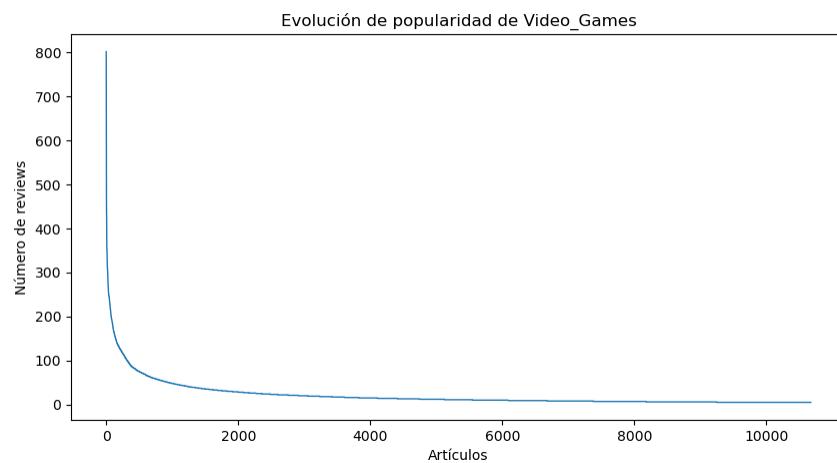


Figura 36: Evolución de la popularidad de Videogames

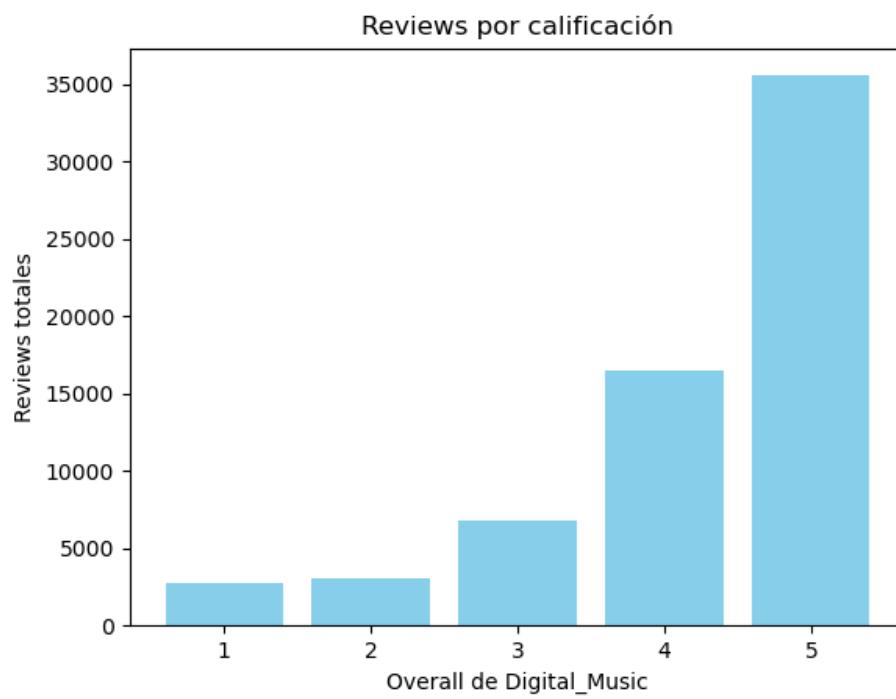


Figura 37: Reviews por nota para Digital Music

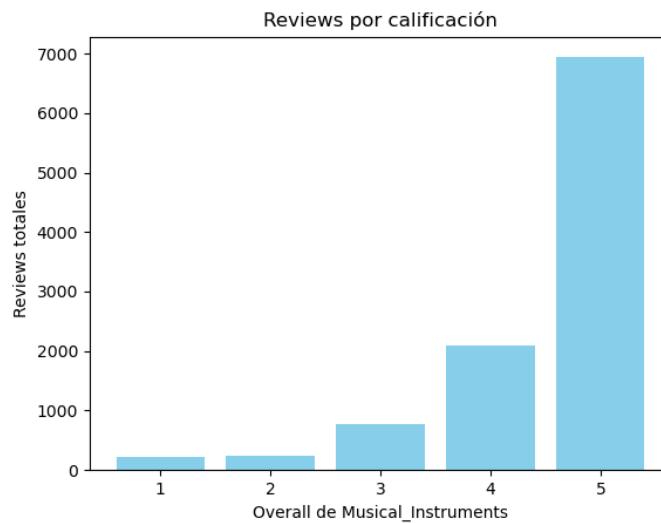


Figura 38: Reviews por nota para Musical Instruments

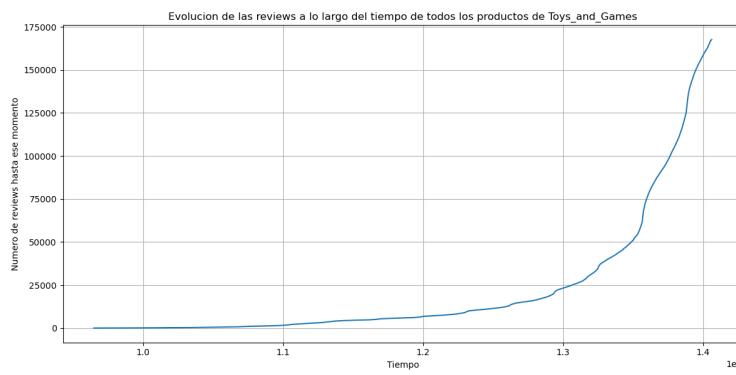


Figura 39: Evolución de las reviews con el tiempo para Toys and Games

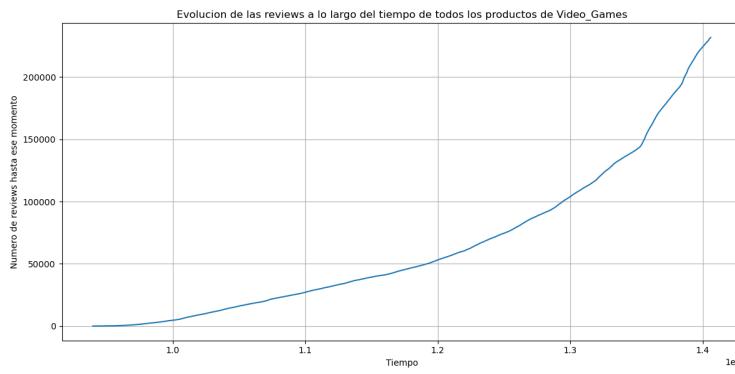


Figura 40: Evolución de las reviews con el tiempo para Videogames



Figura 41: Nube de palabras para Musical Instruments



Figura 42: Nube de palabras para Toys and Games

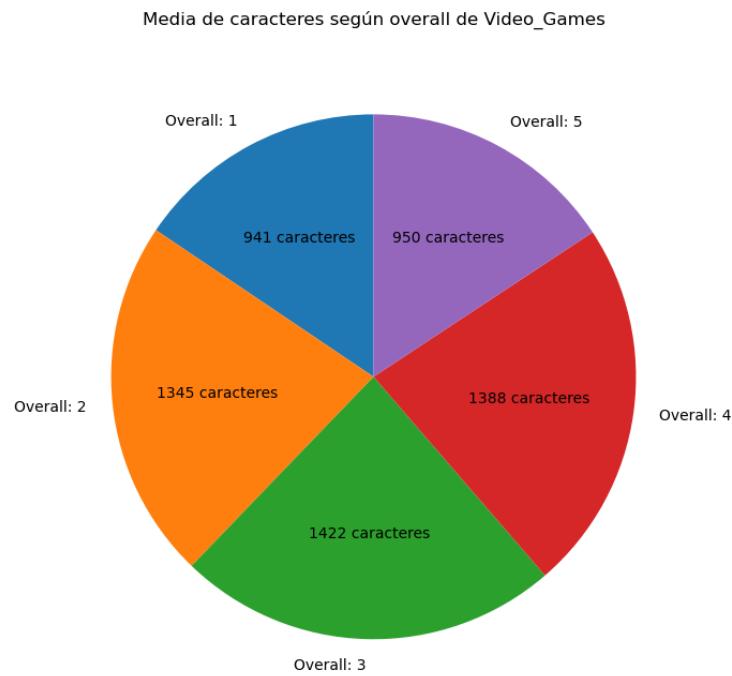


Figura 43: Pie Plot media de caracteres por overall para Videogames

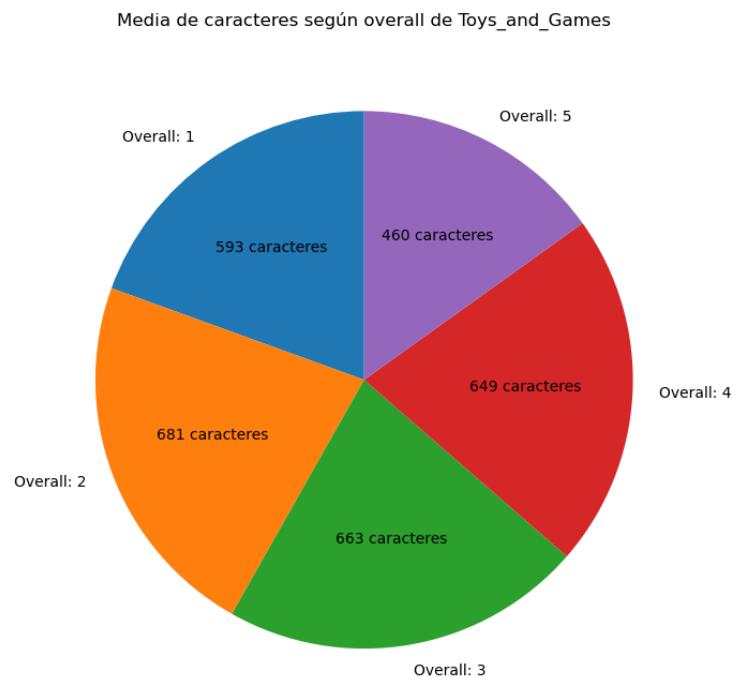


Figura 44: Pie Plot media de caracteres por overall para Toys and Games