Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2018)

Laboratorio 4

Objetivo

- Aprender a modelar correctamente una base de datos relacional.
- Realizar consultas a la base de datos construida mediante SQL.
- Utilizar Python para realizar consultas y modificaciones a la base de datos.

Entrega

- **Lenguaje a utilizar:** Python 3.6
- **Lugar:** repositorio GitHub. Recuerde incluir todo en una carpeta de nombre **L04**.
- **Fecha límite de entrega: miércoles 10 de octubre, a las 23:59 hrs.**
- **Formato de entrega:** dos archivos con la solución propuesta, uno en formato **.py** y el otro en **.ipynb**. Deben ser exactamente iguales (el ayudante revisará cualquiera de ellos). Adicionalmente, incluya un archivo **README.md** con información útil para el ayudante corrector.

Introducción

Decides convertirte en desarrollador de aplicaciones móviles, pero antes de comenzar a crearlas, necesitas conocer a la competencia. Para esto, has buscado y encontrado una base de datos de aplicaciones móviles. Quieres usar esta información para tomar algunas decisiones de diseño. Algunas de estas decisiones son:

- Versión de Android a utilizar
- Hacer una aplicación gratuita o de pago

- Frecuencia recomendada de las actualizaciones
- Qué aplicación recomendarías a algún usuario

Para conocer la respuesta a esta y otras preguntas, decides hacer un programa que consulte a la base de datos utilizando Python y SQL.

Instrucciones

Este laboratorio se divide en tres partes: procesamiento de la información, modelación y consultas. Además, se incluye una sección inicial con información de la base de datos. Cada una de ellas se describe a continuación.

Base de datos

Archivos CSV

Los archivos CSV (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: `Chile,Perú,Argentina,España,Brasil...)` y las filas por saltos de línea.[1]. En caso que algún elemento de una columna posea coma en el texto, este elemento será encerrado entre comillas, por ejemplo:

```
PLANK!,GAME,4.7,7196,38M,"500,000+",Free,0,Everyone,Arcade,"July 17, 2018",1.0.2,4.3 and up
```

En este ejemplo el "500,000+" y "July 17, 2018" poseen comas y por lo tanto, en el archivo se le pone comillas para diferenciar la coma del texto y la coma como separador. Para la lectura de estos archivos se recomienda utilizar alguna librería como **Pandas** o **CSV**.

Archivos entregados

Para este laboratorio, dispondrás de 2 archivos CSV con la información inicial del sistema. El primero se llama *googleplaystore.csv*, el cual contiene la información de cada aplicación. Sus columnas son las siguientes:

- **App**: nombre de la aplicación

[1] Cita de wikipedia

- **Category**: categoría de la aplicación. Entre estas se encuentran *ART_AND_DESIGN*, *FAMILY*, *GAME*, etc.
- **RatingOverall**: *rating* de la aplicación
- **Reviews**: número de comentarios que tiene la aplicación. Esta columna debe ser ignorada. En otro archivo estan todos los comentarios y esa cantidad será la utilizada.
- **Size**: tamaño de la aplicación
- **Installs**: indicador que representa cantidad de descarga de la aplicación.
- **Type**: indica si es pagada o gratuita.
- **Price**: precio de la aplicación
- **Content Rating**: grupo de edad a la que va dirigida la aplicación.
- **Genres**: género de la aplicación, se puede considerar como una sub-categoría y cada aplicación puede tener más de un género. En ese caso, cada género está dividido por un punto y coma (";").
- **Last Updated**: fecha de la última vez que fue actualizada la aplicación
- **Current Ver**: versión de la aplicación.
- **Android Ver**: versión de *Android* requerida por la aplicación.

El segundo archivo se llama *googleplaystore_user_reviews.csv* que contiene diferentes comentarios de los usuarios para cada aplicación. Las columnas son:

- **App**: nombre de la aplicación.
- **Translated_Review**: Comentario de la aplicación. Todas traducidas al ingles.
- **Sentiment**: Indica si el comentario es positivo, negativo o neutral.
- **Sentiment_Polarityws**: puntaje asociado a la polaridad del comentario, valor entre -1 y 1. Esta columna debe ser ignorada.
- **Sentiment_Subjectivity**: puntaje asociado al sentimiento entregado en el comentario. Esta columna debe ser ignorada.

Procesamiento

Dentro de los archivos, hay varias aplicaciones o comentarios con valores `NaN` los cuales deberán eliminar antes de pasar a la siguiente etapa.

Modelación

Luego del procesamiento, deberán pasar la información a un base de datos relacional en SQL. Para esto deben crear múltiples entidades y relaciones que **aseguren las restricciones de integridad**. Es requisito que existan al menos **3 tablas que representen entidades y 2 que representen relaciones entre entidades**, y todas las entidades posean un **id numérico único** de manera que las comparaciones para realizar *joins* entre tablas se haga solamente basado en estos ids. El modelo debe incluir la misma información entregada en los .CSV.

main.py

Para este laboratorio, se les adjunta un archivo de nombre *main.py* y *main.ipynb* que deberán completar con la solución. El archivo contiene el formato esperado de las consultas descritas en la siguiente sección. Es obligatorio seguir el formato allí descrito, pero queda a su criterio el desarrollo de cualquier otra función o clase que sea necesaria. Podrán notar del archivo, que este ya incluye ejecuciones de las consultas y se basa en que estas retornen la solución en el formato solicitado. En caso de no respetar dicho formato, no seré evaluada la funcionalidad.

Ediciones a la base de datos

Antes de las consultas, su desarrollo del laboratorio debe incluir 4 funciones para agregar, editar y eliminar información de su base de datos. Las funciones son:

- **add_app(app_data)**: esta función recibe un diccionario con la información de la aplicación. El formato del diccionario es:

Llave del diccionario	Valor del diccionario	Condición del valor
name	nombre de la aplicación	Debe ser un `string`.
category	categoría de la aplicación	Debe ser un `string`.
rating	rating de la aplicación.	Debe ser un número > 0.
size	tamaño de la aplicación	Debe ser un número > 0.
price	precio de la aplicación.	Debe ser un número ≥ 0.
version	versión de la aplicación	Deber ser un número > 0.
android	versión de *android* de la aplicación	Deber ser un número > 0.
genres	géneros de la aplicación	Debe ser una lista de `string`.

Cuando se agrega la aplicación, debe suponer que esta tiene 0 comentarios, 0 instalaciones y que la última actualización fue cuando se agregó la aplicación a la base de datos. Para la lista de géneros, si algún género no existe, este debe ser agregado a la base de datos. En caso que un valor del diccionario no sea correcta, debe imprimir en consola "No es posible agregar la aplicación" y retornar `None`.

- **add_comment(app_name, text, sentiment)**: agregar un nuevo comentario de una aplicación. Debe recibir el nombre de la aplicación, un texto y un sentimiento (positivo, neutro o negativo). Puede suponer que siempre va a existir el nombre de la aplicación, *text* será un *string*. Esta función debe retornar `None`.

- **download_app(app_name)**: descarga una aplicación. Esto incrementa en 1 el número del atributo descargas. Puede suponer que siempre va a existir el nombre de la aplicación. Esta función debe retornar `None`.

- **delete_app(app_name)**: elimina toda la información asociada a la aplicación con el nombre ingresado. Puede suponer que siempre va a existir el nombre de la aplicación. Esta función debe retornar `None`.

Consultas

Las consultas mínimas que debe soportar su sistema son:

- **get_info(app)**: dado el nombre de una aplicación (app), retorna la información de dicha aplicación en forma del diccionario. En particular, la información retornada debe seguir el siguiente formato en términos de *key*:

```
{
    "name": "nombre de la aplicación",
    "category": "categoría de la aplicación",
    "ratingOverall": "rating de la aplicación.",
    "reviews": "cantidad de comentarios",
    "size": "tamaño de la aplicación",
    "installs": "cantidad de instalaciones",
}
```

En caso que el nombre de la aplicación no exista, debe retornar el siguiente diccionario:

```
{
    "error": "La aplicación {nombre de la aplicación} no existe en la base de datos"
}
```

- **best_by_genre(n, genre)**: esta consulta recibe un número entero (n) y un género (genre) de la base de datos. Imprime en consola los nombres de las n mejores aplicaciones de dicho género. En caso de que el genero no exista, debe imprimir en consola "Este género no existe".

- **count_by_version(date_1, date_2)**: dadas dos fechas (date_1, date_2), deben imprimir en consola cuantas aplicaciones por versión de Android hay y cada versión, indicar la categoría mejor evaluada. Para esta última parte, deben considerar el *rating* promedio de las aplicaciones de dicha categoría para encontrar la mejor evaluada. Las fechas vendrán como string y su formato será "YYYY-MM-DD"".

- **price_of_the_best_by_genre(n, genre)**: dado un número entero (n) y un género (genre), debe retornar el precio promedio de las n aplicaciones mejor evaluadas que posean dicho género. En caso de que no exista ese género, debe retornar -1. Esta consulta **debe ser realizada completamente con SQL**. Se espera que ninguna acción (aparte del `return`) sea realizada con comandos propios de Python, solo con comandos de SQLite.

- **recommend(genre, size)**: Dado un género (genre) y un número (size), debes recomendar hasta 5 aplicaciones de dicho género que tengan un tamaño menor al indicado en size. Para recomendar, debes ordenar las aplicaciones de posean dicho género en base a la cantidad de comentarios con sentimiento positivo. Esta consulta debe retornar una lista con los nombres de las aplicaciones recomendadas.

- **need_update_(app)**: dado un `string` con el nombre de una aplicación (app), se debe calcular el fecha promedio de la última actualización de las aplicaciones que pertenecen a dicha categoría y que

poseen una versión de Android igual o superior a la aplicación dada. La función retorna True o False de acuerdo a si la fecha de la última actualización de la aplicación es más antigua que el promedio de su categoría. En esta función, **debe ser realizada completamente con SQL**, toda la búsqueda y filtrado debe realizarse mediante consultas SQL, puede utilizar Python para calcular el promedio y retornar lo correspondiente.

- **app_with_more_income()**: se debe retornar el nombre de la aplicación que ha recibido más ingresos. En caso de empate, se deben retornar hasta cinco nombres. Haga todos los supuestos que estime conveniente y especifíquelos en el informe.

Informe

Para este laboratorio se les pedirá escribir un informe que incluya los siguientes puntos:

- Diagrama del modelo de la base de datos. Algunas aplicaciones *online* para dibujar su modelo son: **Draw.io** y **ERDPlus**.
- Justificación de por qué el modelo escogido es el adecuado para el problema y las consultas planteadas. Debe basarse en los criterios de consistencia para la justificación

El informe puede escribirse en formato Markdown y puede incluirse como una sección del README. Opcionalmente, se premiará con hasta 5 décimas en la nota final del Laboratorio para los que realicen su informe en LaTeX. Para optar a este beneficio, se debe incluir en el repositorio el código **.tex** con todo lo necesario para que el código compile y el archivo **.pdf**. De esta forma, la nota máxima que pueden obtener en este laboratorio es un 8.0 (considerando el bonus por el informe y la bonificación a criterio del corrector).

Avance parcial

Para la revisión de avance, se espera que las parejas tengan al menos desarrollado lo siguiente

- modelo de datos completo: diagrama y definición de tablas en una base de datos SQLight.
- datos cargados en la base SQLight, de acuerdo al modelo previamente definido.
- Una manera de verificar el contenido de las tablas.
- Una consulta cualquiera de las requeridas.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.