Parcial 2: Programación

Muchos éxitos!
Este es el primer parcial de la asignatura programación, tienen ocho días para desarrollar el parcial. Cada estudiante se asignará una letra (entre la A y la I), la cual **NO** podrá cambiar. Asociada a esa letra debe desarrollar los 10 ejercicios asignados. Estos deben ser desarrollados en una rama diferente a la principal de un repositorio de GitHub (Junto con los archivos de datos a leer por el programa) y al estar en la versión definitiva hacer un Merge a la rama principal. Deben tener un archivo README usando comandos eseciales de Markdown para explicar como se resolvió el examen. El proyecto debe tener por lo menos 3 commits espaciados al menos un día entre sí (Puede tener más). El enlace del repositorio se envía al correo secheverri@uniquindio.edu.co antes de la fecha límite.
Puede usar cualquier librería. Cada estudiante deberá sustentar su parcial. Hay asesoría durante el desarrollo del parcial. La asignación de los ejercicios es:

Estudiante	A	B	C	D	E	F	G	H	I
Ejercicio 1	†	†	†	†	†	†	†	†	†
Ejercicio 2	†	†	†						
Ejercicio 3				†	†	†			
Ejercicio 4							†	†	†
Ejercicio 5	†			†				†	
Ejercicio 6	†				†				†
Ejercicio 7		†		†	†				
Ejercicio 8		†				†		†	†
Ejercicio 9			†				†		
Ejercicio 10			†			†	†		
Ejercicio 11	†	†	†	†	†	†	†	†	†
Ejercicio 12	†	†	†	†	†	†	†	†	†
Ejercicio 13	†			†			†		
Ejercicio 14		†			†			†	
Ejercicio 15			†			†			†
Ejercicio 16	†	†	†	†	†	†	†	†	†
Ejercicio 17	†	†	†	†	†	†	†	†	†
Ejercicio 18	†	†	†	†	†	†	†	†	†

Cuadro 1: Asignación de examenes

1. Importar datos

1. Construya una clase que pueda recibir como argumentos **al momento de definirla** el nombre de uno o varios archivos y opcionalmente la ubicación de cada uno. Si no se ingresa la ruta de algun archivo el programa supone que es la ruta actual y lee todos

los archivos de datos. Si no se le ingresa el nombre de ningún archivo el programa debe generar la base de datos.

2. Las bases de datos que lee el programa las debe conseguir (O modificar alguna existente dividiendola en varios archivos) y deben tener varias columnas con el mismo nombre. Al leer las bases de datos se genera una única tabla de datos con las columnas comunes y los datos de todos los archivos. **La tabla final no puede tener filas que tengan en común más de tres valores (Solo se dejaría uno de ellos por considerarse datos repetidos)**

3. Las bases de datos que lee el programa son solo dos y las debe conseguir o generar, una de ellas tiene una columna con valores únicos. La segunda tabla hace referencia a elementos de esa primera tabla por medio de esos índices. Guarde las tablas de forma que el programa sea capaz de unir ambas tablas de forma correcta en una sola. Ver estructura de tablas en Fig. 1 **Si lo resuleve usando SQL no debe hacer los ejercicios de procesar datos**

We currently have a `team` table:

id	name	headquarters
1	Preventers	Sharp Tower
2	Z-Force	Sister Margaret's Bar

And a `hero` table:

id	name	secret_name	age	team_id
1	Deadpond	Dive Wilson	null	2
2	Rusty-Man	Tommy Sharp	48	1
3	Spider-Boy	Pedro Parqueador	null	1

Figura 1: Estructura de tablas para numeral 3

4. Como en el ejercicio 3, pero en lugar de unir todas las columnas en una sola tabla, el programa le muestra al usuario las columnas de las dos tablas y el usuario selecciona las columnas que necesita. La tabla que se guarda debe tener solo esas columnas. **Si lo resuleve usando SQL no debe hacer los ejercicios de procesar datos**

Parcial 2: Programación

2. Procesar datos

5. La clase debe tener un método que detecte automátiamente en las columnas con valores tipo str los datos que se identifican como diferentes de forma errada por cuestión de tildes o mayúsculas/minúsculas y homogeneizar los valores. Además le debe mostrar al usuario los valores únicos y permitirle decir si desea unir o no algunos de esos valores.

6. La clase debe tener un método que analice la Gausianidad, curtosis y sesgo de cada columna numérica y le dé una gráfica Q-Q de cada columna numérica.

7. La clase debe tener un método que reciba condiciones para los datos numéricos y filtre los datos de la tabla con base en esa condición

8. La clase debe tener un método que reciba condiciones para los datos tipo cadena y filtre los datos de la tabla con base en esa condición

9. La clase debe tener un método que tome las variables numéricas y haga un gráfico de caja y bigotes, diciendo cuáles datos son outliers

10. La clase debe tener un método que genere una matriz que tenga en cada elemento i, j la correlación entre la columna i y la columna j

11. La clase debe tener un método que calcule los coeficientes de regresión lineal para dos columnas elegidas por el usuario. (Si hace regresión multi-lineal gana puntos adicionales, si la hace multi-lineal con regularización, gana aún más puntos adicionales)

3. Graficar

12. Grafique el resultado del numeral anterior. En scatter los valores de la tabla y en línea la regresión lineal.

13. Realice un método o una función que al recibir una función matemática definida a partir de una lambda, realice su integral, su derivada y las grafique

14. Realice un método o una función que al recibir una función matemática definida a partir de una lambda, encuentre los cortes con el eje x, grafique la función y ponga en scatter los puntos solución.

15. Implemente un método que al recibir una función matemática en dos variables definida a partir de una lambda, y el nombre de dos columnas de la tabla, grafique en 3D o en contornos esa gráfica

Parcial 2: Programación

4. Requisitos generales

16. Defina y use al menos dos decoradores
17. Use *args y **kwargs
18. Documente TODAS las funciones y métodos

Si se pide que construya una clase que pueda hacer algo, debe implementar un ejemplo de uso en el código.
El código puede ser en formato .py o .ipynb