Gestión de la Información en la Web
Curso 2024-25
Práctica 4 – Acceso a Bases de Datos

Fecha de entrega: domingo 27 de octubre de 2024

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero `pr4.py`. El esqueleto de este fichero se puede descargar del Campus Virtual.

Lenguaje de programación
Python 3.11 o superior.

Calificación
Se medirá la corrección mediante tests de unidad. Además de la corrección, se valorará la **calidad, concisión y claridad del código**, la incorporación de **comentarios** explicativos, su **eficiencia** tanto en tiempo como en memoria y la puntuación obtenida en Pylint.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

Considerar una base de datos sobre diferentes índices bursátiles formada por las siguientes tablas. Debéis utilizar **exactamente estos nombres de tablas y columnas**. Para simplificar, ninguno de estos nombres lleva espacios ni tildes.

Tabla «datos_generales»: información general de distintas acciones de diferentes mercados bursátiles.

- **ticker** (clave primaria): cadena de texto
- **nombre**: cadena de texto
- **indice**: cadena de texto
- **pais**: cadena de texto

Tabla «semanales_IBEX35»: datos históricos semanales de las acciones del IBEX35.

- **ticker** (valor que hace referencia al campo *ticker* de la tabla «*datos_generales*»): cadena de texto
- **fecha**: cadena de texto
- **precio**: número decimal

En esta tabla la clave primaria está formada por el **ticker** y la **fecha**.

1. Crear las tablas y añadir las filas [5pt]

Crear dos funciones Python para crear la base de datos y rellenarla de datos a partir de ficheros CSV:

1. La función `crear_bd(db_filename)` almacena la base de datos en el fichero `db_filename` y crea las dos tablas indicadas. Los nombres de las tablas y de las columnas deben ser exactamente los que aparecen en los listados anteriores. Con respecto a los tipos de datos, debéis decidir qué tipos de SQLite son los más adecuados para cada columna. Además, debéis detectar y configurar las claves primarias y claves foráneas (si existen) de cada tabla.
2. La función `cargar_bd(db_filename, tab_datos, tab_ibex35)` recibe la ruta del fichero con la base de datos ya creada además de las rutas a los ficheros CSV con los datos de las tablas: «tab_datos» contendrá los datos generales y «tab_ibex35» contendrá datos históricos semanales del IBEX35[1]. En el caso de la tabla de datos históricos del IBEX35, será necesario procesar los datos en bruto procedentes del fichero CSV para almacenar las fechas en formato `YYYY-MM-DD HH:MM` inspirándonos en el estándar ISO-8601[2], es decir, la fecha «`19/08/2022 17:36`» debe ser transformada en «`2022-08-19 17:36`» antes de almacenarla en la tabla. Esto es importante porque una de las consultas que realizaremos necesita ordenar los resultados por fecha. **Importante:** los ficheros pasados como parámetros se deben leer utilizando expresamente la codificación UTF-8.

[1] Podéis encontrar ficheros CSV de ejemplo en el Campus Virtual: `Tabla1.csv` y `Tabla2.csv`
[2] `https://es.wikipedia.org/wiki/ISO_8601`

2. Consultar la base de datos [5pt]

Escribir 4 funciones para consultar la base de datos y **devolver los resultados como una lista de tuplas**. Estas funciones deben abrir una conexión a la base de datos y cerrarla antes de terminar, para no dejar recursos ocupados innecesariamente. Todas las funciones toman como primer parámetro la ruta del fichero donde está almacenada la base de datos.

Comentario general: los apartados que realizan consultas a la BD requieren devolver los resultados ordenados utilizando un único criterio. En caso de empates no debéis aplicar más criterios de ordenación, sino que debéis dejar el orden natural de la BD.

1. La función `consulta1(db_filename, indice)` devuelve una lista de tuplas `(ticker, nombre)` de todas las acciones que componen el `indice` pasado como parámetro. Los resultados se deben devolver ordenados de manera descendente por ticker.

   Ejemplo:

   ```
   >>> consulta1('bolsa.sqlite3', "Nasdaq 100")
   [('AAPL', 'APPLE INC'),
    ('ABNB', 'AIRBNB RG-A'),
    ('ADBE', 'ADOBE SYSTEMS INC'),
    ('ADI', 'ANALOG DEVICES'),
    ...
   ]
   ```

2. La función `consulta2(db_filename)` devuelve una lista de tuplas `(ticker, nombre, precio máximo)` de las distintas acciones del IBEX35 según los datos históricos. Los resultados se deben devolver ordenados por nombre ascendente.

   Ejemplo:

   ```
   >>> consulta2('bolsa.sqlite3')
   [('ANA', 'ACCIONA', 207.0),
    ('ACX', 'ACERINOX SA', 12.68),
    ('ACS', 'ACS', 26.6),
    ...
   ]
   ```

3. La función `consulta3(db_filename, limite)` devuelve una lista de tuplas

   `(ticker, nombre, precio promedio, diferencia entre el precio máximo y mínimo)`

   de las distintas acciones del IBEX35 según los datos históricos. Únicamente se deben mostrar aquellas acciones cuyo precio promedio sea superior a `limite`. Además, los resultados se deben devolver ordenados por precio promedio descendente.

   Ejemplo:

   ```
   >>> consulta3('bolsa.sqlite3', 10)
   [('ANA', 'ACCIONA', 173.75098039215692, 67.69999999999999),
    ('AENA', 'AENA', 135.57058823529408, 46.54999999999998),
    ('PHM', 'PHARMA MAR', 62.54470588235293, 25.060000000000002),
    ...
   ]
   ```

4. La función `consulta4(db_filename, ticker)` devuelve una lista de tuplas

`(ticker, fecha, precio)`

de las acciones del `ticker` pasado como parámetro según el histórico de valores semanales del IBEX35. Los resultados deben aparecer ordenados por fecha de más reciente a más antiguo. **Importante**: el campo `fecha` debe mostrar únicamente el día *y no la hora*, por lo que deberéis utilizar alguna función de manejo de fechas[3] para extraer la información que os interesa.

Ejemplo:

```
>>> consulta4('bolsa.sqlite3', 'ACS')
[('ACS', '2022-09-30', 23.13),
 ('ACS', '2022-09-23', 22.13),
 ('ACS', '2022-09-16', 22.92),
 ('ACS', '2022-09-09', 23.01),
 ...
]
```

[3] `https://www.sqlite.org/lang_datefunc.html`