CEI.

Curso 2023/2024.

PRUEBA: Programación.
Evaluación: Tema 10 y tema 11
Fecha: 06/06/2024.

Nombre y Apellidos:

Grupo: Mañana

Normativa:

1. Entregar un fichero zip con el nombre **1ºAPELLIDO_2ºAPELLIDO_NOMBRE.zip** donde irá el proyecto exportado.
2. Si el proyecto no se importa correctamente la nota será un 0.
3. Si el proyecto no compila la nota será un 0.
4. Que el programa del resultado correcto no implica tener la máxima nota.

Se desea implementar una aplicación que gestione un centro educativo privado. La estructura de datos y su relación se encuentra en los siguientes ficheros:

- script.sql
- Diagrama.png
- https://drawsql.app/teams/ignacio-programacion/diagrams/exament10

Tambien teneis el fichero poblar_base_de_datos.sql con INSERT para tener algunos datos de prueba (no quiere decir que sean suficientes).

El cliente quiere priorizar la optimización de consultas, seguridad e integridad de datos, por lo que **tendrá muy en cuenta** en el desarrollo que realicemos.

Solo se permiten sentencias simples, ya sea para obtener datos, modificarlos o eliminarlos. Es decir, para un SELECT por ejemplo solo se puede usar el formato SELECT id_curso FROM Curso WHERE nombre='Curso de matemáticas', no se podrán usar sentencias como JOIN, DISTINCT, COUNT, etc.

Tened en cuenta todas las posibles casuísticas que se pueden dar a la hora de resolver los apartados.

El cliente quiere implementar las siguientes funcionalidades:

1. Método para obtener un Map de estudiantes con ID y nombre a partir de una lista de ids de estudiantes: **(0.5p)**
   ```
   public Map<Integer, String> obtenerEstudiantesPorID(List<Integer> estudiantesIDs)
   ```

2. Método para actualizar los emails de estudiantes a partir de un hashtable de id de estudiante y email nuevo: **(0.5p)**
   ```
   public boolean actualizarEmailEstudiantes(Map<Integer, String> estudiantes)
   ```

3. Método para obtener una lista con todas las clases donde un profesor da un curso concreto: **(0,5p)**
   ```
   public List<Clase> obtenerClaseDeCursoPorProfesor(int idProfesor, String nombreCurso)
   ```

4. Método para obtener los estudiantes que reciben clases para un curso concreto, comprobando que el profesor da ese curso: **(1p)**
   ```
   public List<Estudiante> obtenerEstudiantesPorProfesorYCurso(int idProfesor, int idCurso)
   ```

5. Método que intercambia dos profesores de curso, por ejemplo, Carlos pasa de dar el curso de matemáticas básicas a dar el de historia universal y Pedro pasa de dar historia universal a dar el de matemáticas básicas: **(2.5p)**

```
public void intercambiarProfesoresDeCurso(int idProfesor1, int
idProfesor2, int cursoProfesor1, int cursoProfesor2)
```

6. Método que ejecuta el procedimiento *consultar_datos* almacenado en la BBDD:
   `{call` *consultar_datos*`(?, ?)}`
   `public static void llamarProcedimiento(int estudianteId, int cursoId, int profesorId, int claseId)` **(1p)**

7. Método que busca un número "n" de clases aleatorias y devuelve la lista de estudiantes que reciben cursos en dichas clases (no deben repetirse estudiantes): **(1p)**
   `public List<Estudiante> buscarEstudiantesEnClasesAleatorias(int n)`

8. Método que busca un curso y comprueba que dicho curso no tenga más de "n" alumnos, en caso de tenerlos deberá de desdoblar dicho curso en el número de clases pertinentes (por ejemplo si tiene 20 alumnos y el máximo de alumnos es 9 entonces serán un total de 3 clases, dos con 9 alumnos y una con dos alumnos)

   Para cada una de las nuevas clases tendrá que asignar profesores que no estén en ningún curso en dicho día y hora y que pertenezcan al departamento. En el caso de no tener profesores se deberá de avisar de ello por consola y mantener la integridad de la base de datos.

   Además, ha de tener en cuenta que las clases que se asignen adicionales para los grupos de dicho curso en dicho horario tiene que estar libres, es decir, que la clase en dicho horario no haya un profesor que la utilice para dar un curso y se le asignará como nombre el de la clase original. En el caso de no tener clases libres para ese día y esa hora se deberá de avisar de ello por consola y mantener la integridad de la base de datos.

   Tened en cuenta todas las relaciones que existen para insertar y/o modificar todos los datos necesarios en sus respectivas tablas: **(3p)**
   `public void dividirCurso(int idCurso, int n)`