1. Vistas de Ocupación

1. Crear una vista `V_OCUPACION_ASIGNADA` que reúna las tablas de sedes, examen y asistencia y devuelva el código de sede, su nombre, código de aula, fecha de examen y número de estudiantes asignados a un examen, agrupando por dicha sede, su nombre, código de aula y fecha de examen.
2. Crear una vista `V_OCUPACION` que reúna las tablas de sedes, examen y asistencia y devuelva el código de sede, su nombre, código de aula, fecha de examen y número de estudiantes que han asistido a un examen (atributo ASISTE = 'SI'), agrupando por dicha sede, su nombre, código de aula y fecha de examen.
3. Crear una vista `V_VIGILANTES` que reúna las tablas de sedes, examen y vigilancia y devuelva el código de sede, su nombre, código de aula, fecha de examen y número de vigilantes que han vigilado un examen, agrupando por dicha sede, su nombre, código de aula y fecha de examen.

2. Paquete PK_OCUPACION

Implementar un paquete con las siguientes funcionalidades:

1. Implementar una función denominada `OCUPACION_MAXIMA` que reciba como argumento el código de sede y de aula y devuelva el número máximo simultáneo de personas que han estado presente en ella. Hay que tener en cuenta los diferentes exámenes y además que el número de personas es igual al número de estudiantes, más los vocales presentes en ella.
2. Implementar una función denominada `OCUPACION_OK` que devuelva un booleano a TRUE si todos los exámenes **aún no realizados** tienen un número de estudiantes asignados a un aula inferior o igual al atributo `Capacidad_Examen` y el número total de personas (`alumnos + vocales`) no supera al atributo `Capacidad`.
3. Implementar una función denominada `VOCAL_DUPLICADO` que reciba como argumento el identificador de un vocal y devuelva `TRUE` si dicho vocal está asignado a más de un examen en la misma franja horaria. Como es de esperar esto es algo que no debería ocurrir.
4. Implementar una función denominada `VOCALES_DUPLICADOS` que devuelva `TRUE` si alguno de los vocales está asignado a más de un examen en la misma franja.
5. Implementar una función denominada `VOCAL_RATIO` que recibe un número entero y que devuelva un booleano a TRUE si en todos los exámenes aún no realizados hay el número indicado (o menos) de alumnos por cada vigilante.

3. SEGURIDAD

1. Crear un paquete con 2 procedimientos `PR_CREA_ESTUDIANTE` y `PR_CREA_VOCAL` que reciban el identificador del individuo y que tenga dos

argumentos de salida que corresponderán el primero al nombre del usuario creado en Oracle y el segundo a la contraseña generada (puedes utilizar la función `STRING` del paquete `DBMS_RANDOM`). Estos procedimientos además deberán asignar todos los roles, permisos que sean necesarios. No dudes en modificar el esquema de la base de datos si fuera necesario.

4. Trigger

1. Queremos implementar (mediante un trigger denominado `TR_BORRA_AULA`) el borrado de un aula (mediante una sentencia `DELETE` sobre la tabla `AULA`) que cumpla las siguientes condiciones:
    a. El borrado de un aula no es posible si ya se ha realizado un examen, o bien si hay planificado un examen antes de las próximas 48 horas.
    b. Si no se ha realizado examen ni hay planificado uno en las próximas 48 horas el borrado del aula implica el borrado de los exámenes planificados en la misma.

5. Procedimientos adicionales

1. Implementar un procedimiento denominado `DESPISTE` que reubique a un estudiante que por error se ha presentado a un examen en una sede que no es la que le corresponde por su Centro. El procedimiento recibe como argumento un identificador de estudiante (DNI) y un identificador de examen (los atributos necesarios, por ejemplo, Fecha y hora, Aula y sede).
    a. Este procedimiento únicamente funcionará si queda menos de una hora para el comienzo del primer examen al que el alumno se tenga que presentar. Para ello se obtiene la **primera** hora de la tabla `ASISTE` que corresponda con el DNI del alumno y se comprueba la diferencia entre la fecha programada y la hora del sistema. Se comprueba si falta menos de una hora. Si no es así, el procedimiento devolverá una excepción. Recuerda que el tipo DATE almacena tanto la fecha como la hora. Probar con `ASISTE.FechayHora between SYSDATE AND SYSDATE + 1/24`
    b. Si el apartado anterior no ha dado error, el procedimiento debe permitir la asistencia del alumno en la sede nueva. Es decir, debe modificar la fila de la tabla `ASISTE` con el identificador del estudiante, la fecha, el código de la nueva sede y el código del aula pasada como parámetro.
    c. El procedimiento hará lo mismo con el resto de los exámenes de la fecha del sistema, para que el estudiante no tenga que moverse de sede. Para ello hay que recorrer el resto de los exámenes a los que el alumno tiene que presentarse ese día y buscarle un aula libre. Para cada examen se modifica la tabla asiste con los nuevos datos (aula y sede). La diferencia con el apartado anterior es que en el anterior el aula nos la pasan como parámetro, pero en este hay que buscarle un aula disponible. Los exámenes de otras fechas no se modifican.
2. Implementar un procedimiento denominado `MIGRAR_CENTRO`, que recibe el identificador de un centro y el identificador de una sede origen y destino.

a. El procedimiento migrará todos sus alumnos de exámenes de la sede origen a la sede destino.
b. El procedimiento repartirá a los alumnos en las aulas disponibles del nuevo centro sin superar nunca la `Capacidad_Examen` de cada aula.
c. Si no es posible realizar dicha asignación el procedimiento deberá lanzar una excepción.
d. En la implementación de este procedimiento el alumno recibirá tres posibles calificaciones según cómo se implemente (de menos puntuación a más puntuación):
    i. Más baja: Como procedimiento y si la asignación falla deja todo a medio hacer.
    ii. Media: Como procedimiento y si la asignación no es posible deja todo como estaba originalmente.
    iii. Más alta: Como trigger al hacer un `UPDATE` del campo de la sede de un centro (si el trigger falla todo debe quedar como estaba originalmente).