

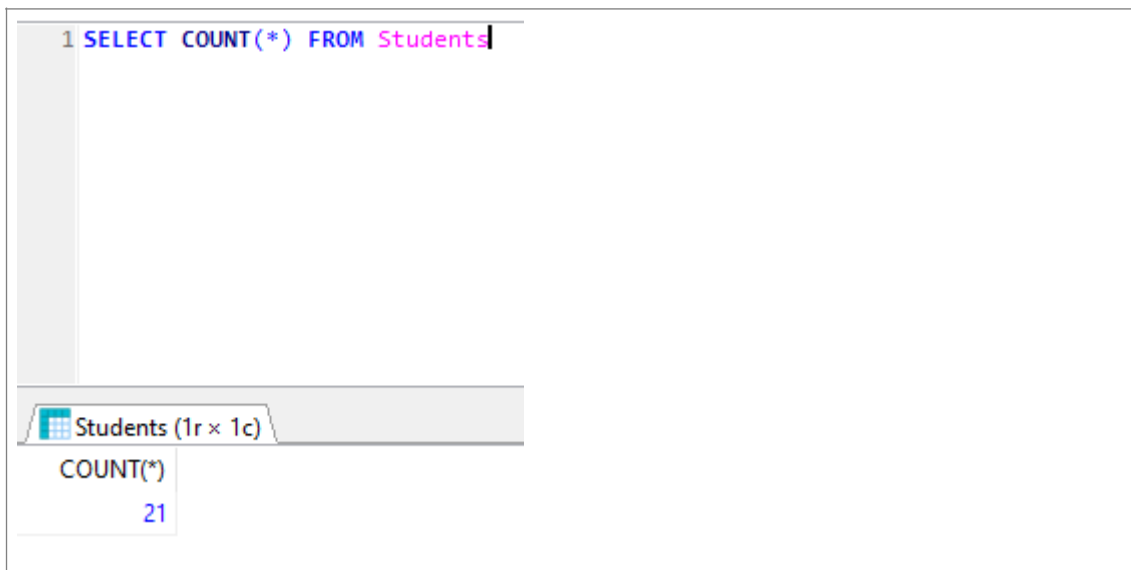
Apellidos, Nombre:  
IP del ordenador:

Grupo:2

Calificación Final: \_\_\_\_\_

**Pregunta 0. (0.5 puntos)**

Utilice HediSQL para abrir la conexión a IISSI\_ROOT con contraseña iissi\$root. Cree una nueva base de datos cuyo nombre sea su UVUS. Ejecute el script SQL proporcionado en la base de datos creada. Para asegurar que todo es correcto ejecute la consulta `SELECT COUNT(*) FROM Students;` y compruebe que el resultado que devuelve es 21.



The screenshot shows the HediSQL interface. At the top, a SQL query is entered in the editor: `1 SELECT COUNT(*) FROM Students;`. Below the editor, a tab labeled "Students (1r x 1c)" is visible. Underneath the tab, the result of the query is displayed in a table with one row and one column. The column header is "COUNT(\*)" and the value is "21".

COUNT(*)
21

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

### Pregunta 1 (0.5 puntos)

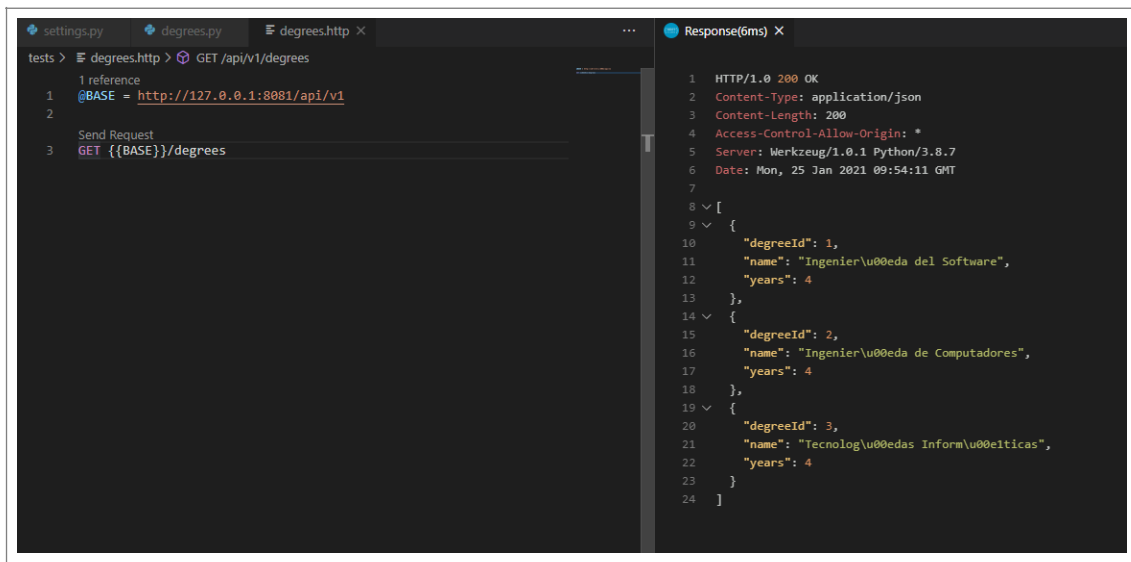
Cree un proyecto Silence y configúrelo para que se conecte a la base de datos creada. Cree y despliegue un endpoint en Silence para la tabla Degrees que devuelva todas las filas de la tabla. Use REST Client para realizar una consulta al endpoint creado.

Según el laboratorio en el que se encuentre, debe configurar además los siguientes parámetros de su proyecto en `settings.py`:

- Puerto de la base de datos (DB\_CONN → port):
  - Laboratorios módulo F: **3308**
  - En otro caso: **3306**
- Puerto de despliegue de la API (HTTP\_PORT):
  - Laboratorios módulo F: **8081**
  - En otro caso: **8080**

Recuerde los siguientes comandos en caso de que sean necesarios:

- Inicialización de proyecto: `silence new <nombre>`
- Puesta en marcha: `silence run` (desde la carpeta del proyecto)



The screenshot shows the REST Client interface with two panes. The left pane shows the request configuration: a GET request to `GET /api/v1/degrees` with a base URL `@BASE = http://127.0.0.1:8081/api/v1`. The right pane shows the response, which is a JSON array of three degree records. The status is `HTTP/1.0 200 OK`.

```
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 200
4 Access-Control-Allow-Origin: *
5 Server: Werkzeug/1.0.1 Python/3.8.7
6 Date: Mon, 25 Jan 2021 09:54:11 GMT
7
8 [
9   {
10    "degreeId": 1,
11    "name": "Ingenier\u00eda del Software",
12    "years": 4
13   },
14   {
15    "degreeId": 2,
16    "name": "Ingenier\u00eda de Computadores",
17    "years": 4
18   },
19   {
20    "degreeId": 3,
21    "name": "Tecnolog\u00edas Inform\u00e1ticas",
22    "years": 4
23   }
24 ]
```

Calificación: \_\_\_\_\_

IISI-1  
Prueba de Laboratorio.  
enero de 2021

Curso 2020-21

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 2. (2 puntos)**

Añada el requisito de información **Publicación**. Una publicación es un artículo publicado en una revista por un profesor. Sus atributos son: el título de la publicación, el profesor que es el autor principal, el número total de autores, la fecha de publicación (día), y el nombre de la revista donde ha sido publicada. Hay que tener en cuenta las siguientes restricciones:

- Un profesor no puede tener varias publicaciones en la misma revista, el mismo día.
- El número de autores debe ser al menos 1 y como máximo 10.
- Todos los atributos son obligatorios a excepción de la fecha de publicación.

```
CREATE TABLE Publications(  
    publicationId INT NOT NULL AUTO_INCREMENT,  
    title VARCHAR(50) NOT NULL,  
    professorId INT NOT NULL,  
    numAuthor INT NOT NULL,  
    pubDay DATE,  
    magazine VARCHAR(50) NOT NULL,  
    PRIMARY KEY (publicationId),  
    FOREIGN KEY (professorId) REFERENCES Professors (professorId),  
    UNIQUE (professorId, pubDay),  
    CONSTRAINT invalidAuthors CHECK (0 < numAuthor < 11)  
);
```

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 3. (1 punto)**

Cree y ejecute un procedimiento almacenado llamado pInsertPublications() que cree las siguientes publicaciones:

- Publicación titulada "Publicación 1" del profesor con ID=1, con 3 autores, publicada en la revista "Revista 1".
- Publicación titulada "Publicación 2", del profesor con ID=1, con 5 autores, publicada el 1 de Enero de 2018 en la revista "Revista 2".
- Publicación titulada "Publicación 3", del profesor con ID=2, con 2 autores, publicada en la revista "Revista 3".

```
DELIMITER //  
CREATE OR REPLACE PROCEDURE  
  pInsertPublications()  
  BEGIN  
    INSERT INTO publications (title, professorId, numAuthor, pubday, magazine) VALUES  
      ('Publicación 1', 1, 3, NULL, 'Revista 1'),  
      ('Publicación 2', 1, 5, '2018-01-01', 'Revista 2'),  
      ('Publicación 3', 2, 2, NULL, 'Revista 3');  
  END //  
DELIMITER ;
```

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 4. (1 punto)**

Cree un disparador llamado tCorrectAuthors que, al actualizarse una publicación, si el número de autores fuera a pasar a ser más de 10, lo cambie a 10 en su lugar.

```
DELIMITER //
CREATE OR REPLACE TRIGGER tCorrectAuthors
BEFORE UPDATE ON Publications
FOR EACH ROW
BEGIN
    if (NEW.numAuthor > 10) then
        SET NEW.numauthor = 10;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
        'El número de autores no puede ser mayor que 10, se ha establecido autmaticamente el valor 10 como valor predeterminado.';
    END if;
END//
DELIMITER ;
```

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 5. (1 punto)**

Cree un procedimiento almacenado llamado pUpdatePublications(p, n) que actualiza el número de autores de las publicaciones del profesor p con el valor n. Ejecute la llamada a pUpdatePublications(1, 10).

Cree un procedimiento almacenado llamado pDeletePublications(p) que elimina las publicaciones del profesor con ID=p. Ejecute la llamada pDeletePublications(2).

```
DELIMITER //
CREATE OR REPLACE PROCEDURE
pUpdatePublications(p INT, n INT)
BEGIN
    UPDATE publications SET numAuthor = n WHERE professorId = p;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE PROCEDURE
pDeletePublications(p INT)
BEGIN
    DELETE FROM publications WHERE professorId = p;
END //
DELIMITER ;
```

publicationId	title	professorId	numAuthor	pubDay	magazine
1	Publicación 1	1	3	(NULL)	Revista 1
2	Publicación 2	1	5	2018-01-01	Revista 2
3	Publicación 3	2	2	(NULL)	Revista 3

publicationId	title	professorId	numAuthor	pubDay	magazine
1	Publicación 1	1	10	(NULL)	Revista 1
2	Publicación 2	1	10	2018-01-01	Revista 2
3	Publicación 3	2	2	(NULL)	Revista 3

publicationId	title	professorId	numAuthor	pubDay	magazine
1	Publicación 1	1	10	(NULL)	Revista 1
2	Publicación 2	1	10	2018-01-01	Revista 2

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 6. (1 punto)**

Cree una consulta que devuelva el nombre del grado, el nombre de la asignatura, el número de créditos de la asignatura y su tipo, para todas las asignaturas que pertenecen a todos los grados. Ordene los resultados por el nombre del grado. Un ejemplo de resultado de esta consulta es el siguiente:

Resultado #1 (4x13)			
degreeName	subjectName	credits	type
Ingeniería de Computadores	Introducción a la Matematica Discreta	6	Formacion Basica
Ingeniería de Computadores	Redes de Computadores	6	Obligatoria
Ingeniería de Computadores	Teoría de Grafos	6	Obligatoria
Ingeniería de Computadores	Aplicaciones de Soft Computing	6	Optativa
Ingeniería del Software	Diseño y Pruebas	12	Obligatoria
Ingeniería del Software	Acceso Inteligente a la Informacion	6	Optativa
Ingeniería del Software	Optimizacion de Sistemas	6	Optativa
Ingeniería del Software	Ingeniería de Requisitos	6	Obligatoria
Ingeniería del Software	Análisis y Diseño de Datos y Algoritmos	12	Obligatoria
Tecnologías Informáticas	Fundamentos de Programación	12	Formacion Basica
Tecnologías Informáticas	Lógica Informatica	6	Optativa
Tecnologías Informáticas	Gestión y Estrategia Empresarial	12	Optativa
Tecnologías Informáticas	Trabajo de Fin de Grado	12	Obligatoria

```
1 SELECT Degrees.name, subjects.name, subjects.credits, subjects.`type` FROM subjects, degrees WHERE subjects.degreeId = degrees.degreeId;
```

Resultado #1 (13x 4c)			
name	name	credits	type
Ingeniería de Computadores	Introducción a la Matematica Discreta	6	Formacion Basica
Ingeniería de Computadores	Redes de Computadores	6	Obligatoria
Ingeniería de Computadores	Teoría de Grafos	6	Obligatoria
Ingeniería de Computadores	Aplicaciones de Soft Computing	6	Optativa
Ingeniería del Software	Diseño y Pruebas	12	Obligatoria
Ingeniería del Software	Acceso Inteligente a la Informacion	6	Optativa
Ingeniería del Software	Optimizacion de Sistemas	6	Optativa
Ingeniería del Software	Ingeniería de Requisitos	6	Obligatoria
Ingeniería del Software	Análisis y Diseño de Datos y Algoritmos	12	Obligatoria
Tecnologías Informáticas	Fundamentos de Programación	12	Formacion Basica
Tecnologías Informáticas	Lógica Informatica	6	Optativa
Tecnologías Informáticas	Gestión y Estrategia Empresarial	12	Optativa
Tecnologías Informáticas	Trabajo de Fin de Grado	12	Obligatoria



IISI-1  
Prueba de Laboratorio.  
enero de 2021

Curso 2020-21

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 7. (1 punto)**

Cree una consulta que devuelva las tutorías con al menos una cita. Un ejemplo de resultado de la consulta anterior es el siguiente:

tutoringhours (1×3)	
tutoringHoursId	
1	
2	
4	

```
1 SELECT tutoringhours.tutoringHoursId FROM tutoringhours, appointments WHERE tutoringhours.tutoringHoursId = appointments.tutoringHoursId;
```

tutoringhours (3r × 1c)	
tutoringHoursId	
1	
2	
4	

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 8. (1 punto)**

Cree una consulta que devuelva la carga media en créditos de docencia del profesor cuyo ID=1. Un ejemplo de resultado de esta consulta es el siguiente

Resultado #1 (1×1)	
avgLoadProfessor1	
	8,0000

```
1 SELECT AVG(teachingloads.credits) FROM teachingloads WHERE professorId = 1;
```

teachingloads (1r × 1c)	
AVG(teachingloads.credits)	
	8,0000

Calificación: \_\_\_\_\_

Apellidos, Nombre:  
IP del ordenador:

Grupo:2

**Pregunta 9. (1 punto)**

Cree una consulta que devuelva el nombre y los apellidos de los dos estudiantes con mayor nota media, sus notas medias, y su nota más baja. Un ejemplo de resultado de la consulta anterior es el siguiente

students (4x2)			
firstName	surname	avgGrade	minGrade
Daniel	Pérez	6,300000	3,25
Rafael	Ramírez	5,833333	2,50

```
1 SELECT students.firstName, students.surname, AVG(grades.value) avgGrade, MIN(grades.value) minGrade
2 FROM students, grades WHERE students.studentId=grades.studentId
3 GROUP BY students.studentId ORDER BY avgGrade DESC LIMIT 2;
```

students (2r x 4c)			
firstName	surname	avgGrade	minGrade
Daniel	Pérez	6,300000	3,25
Rafael	Ramírez	5,833333	2,50

Calificación: \_\_\_\_\_